Winter 12-12-2020

The Design and Validation of a UAV Propeller Intended for Extremely Low Reynolds Number Environments

Benjamin Hebert
University of Maine, ben.hebert@maine.edu

THE DESIGN AND VALIDATION OF A UAV PROPELLER INTENDED FOR EXTREMELY LOW REYNOLDS NUMBER ENVIRONMENTS

By

Benjamin Hebert

B.S. University of Maine, 2017

A THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Masters of Engineering

(in Engineering Physics)

The Graduate School

The University of Maine

December 2020

Advisory Committee:

Samuel Hess, Professor of Physics, Advisor

Wilhelm Friess, Professor of Mechanical Engineering

Richard Eason, Professor of Electrical Engineering

THE DESIGN AND VALIDATION OF A UAV PROPELLER INTENDED FOR LOW REYNOLDS NUMBER ENVIRONMENTS

By Benjamin Hebert

Thesis Advisor: Dr. Samuel Hess

An Abstract of the Thesis Presented
in Partial Fulfillment of the Requirements for the
Degree of Master of Engineering
(in Engineering Physics)
December 2020

Mars exploration and UAV development have both advanced significantly over the past century, and are now being considered in tandem. Currently needed are UAV propellers that can operate in the Martian atmosphere. Flow will be in the range of Re < 20,000, creating extreme conditions not typically examined. A Blade Element Momentum Theory (BEMT) algorithm is developed using a variety of corrections designed specifically for low Reynolds number and rotational flows. Due to both the simplicity of the basic BEMT formulation, corrections are easy to put in place and often necessary to achieve accurate estimates. Aerodynamic coefficients are determined from XFOIL code, and have questionable accuracy in this regime. To account for this, a correction model is developed by comparing XFOIL results to experimental results of airfoils at low Re. This is all tested against a previous low Re propeller experiment. The results of this comparison are used to adjust the values in the correction, to produce more accurate results for theoretical design

From here, a design philosophy for the propeller is developed using established methods and previous experimental data. High thrust is prioritized, with efficiency being a secondary concern. A hard mach limit of 0.7 is set to avoid major drag penalties, limiting the usable ranges of RPM and radius. Airfoil designs are then examined, based on previous designs, theoretical intuition, and experimental

data. A modified version of the S1223 airfoil is adopted for its favorably high C_{lmax} and high stall angle. From here, optimization can be used to determine the final dimensions of the propeller. The BEMT algorithm is used to create a broad set of data, over a range of design variables, which is then fitted to thrust and efficiency functions using non-linear regression. A Non-Dominated Sorting Genetic Algorithm (NSGA) is well suited to optimizing multiple objective functions with multiple design variables, and thus is adopted to optimize the design. The results of the optimization confirm previously published theories, and produce three possible propeller designs, a high thrust model, a high efficiency model, and a compromise between the two. These designs were then modeled, meshed and simulated using the ANSYS Fluent software suite. BEMT thrust estimates were found to be within an average absolute error of ~41% from the simulated results, while moment was within an average absolute error of ~104%. This discrepancy can likely be attributed to the inaccurate drag data being sent into the BEMT algorithm, and the lack of a method to correct said data. With a procedure established for design and testing, new propellers can be created and verified, likely with improvements in accuracy from the initial estimates.

DEDICATION

Dedicated to Katy, who is kind, smart, beautiful, supportive, wonderful, and so many other adjectives I don't have the space to list.

ACKNOWLEDGEMENTS

There are several people I'd like to thank for their help with this project. I'll start with my three advisors Dr. Samuel Hess, Dr. Wilhelm "Alex" Friess, and Dr. Richard Eason for all of their help, suggestions, and input throughout this process. I'd like to thank Dr. Charles Thomas Hess, who helped start this entire project as well as Dr. Qian Xue and Behrad Zeinali, whose expertise with ANSYS allowed me to finish it. Thank you to Pat Byard for her infinite patience with me messing up administrative tasks and for letting me borrow various office keys. Thank you to James Stevens and Michael DeMaria for their early work on the project and Mackenzie Dunning for helping get the whole thing started. I'd also like to thank all of my friends and family who provided encouragement, support, and the occasional push to get me going.

TABLE OF CONTENT

LIST OF TABLE

LIST OF FIGURES

LIST OF SYMBOLS

a = Mach Number

α = Angle of Attack

α_{CLMAX} = Angle of Maximum Lift

$\alpha_{CL=0}$ Angle of Zero Lift

α_i = Initial Angle of Attack

α_{STALL} = Stall Angle

β = Beta Angle

c = Chord Length

c/R = Chord to Total Radius Ratio

c/r = Chord to Local Radius Ratio

C_d = Drag Coefficient

C_{df} = Friction Drag Coefficient

C_{df0} = Pressure Drag Coefficient

C_{dMAX} = Maximum Drag Coefficient

C_{dMIN} = Minimum Drag Coefficient

C_l = Lift Coefficient

C_{lMAX} = Maximum Lift Coefficient

C_l/C_d = Lift to Drag Ratio

D = Drag

g = Gravity

h = Height

L = Lift

λ = Tip Speed Ratio

M = Mach

μ = Viscosity

N_B = Number of Blades

N_{CRIT} = Critical Number

Ω = Rotational Velocity

P = Power

ϕ = Helix Angle

Q = Moment

r = Local Radius

R = Total Blade Radius

R_{HUB} = Hub Radius

r/R = Local Radius Ratio

Re = Reynolds Number

ρ = Density

T = Thrust

Tu(%) = Turbulence Percentage

t/c = Thickness to Chord Ratio

V_E = Effective Velocity

V_0 = Non-Induced Velocity

w = Induced Velocity

CHAPTER 1

A BRIEF HISTORY OF MARS EXPLORATION AND DRONE USE

Humankind has long been fascinated by our neighbor Mars. From ancient astronomers who first spotted it in the night sky, to Edgar Rice Burrough novels in the early 20th century, the red planet has captured the imagination of countless people eager to explore its canyons and ice caps. The revolution in space travel that occurred in the past 70 years has brought us closer to this dream than ever, with land-based rovers currently exploring the surface. In 2020 NASA is set to launch its first mission to Mars involving an Unmanned Aerial Vehicle (UAV) (Green and Grunsfeld 2013), taking the next step forward in interplanetary exploration.

This thesis project was started on a grant from NASA/Maine Space Grant Consortium in partnership with a collaborator at NASA Ames Center, with its primary focus being the design of a propeller that can be used to lift a UAV in the Martian atmosphere. The project itself is built on data gathered from Mars over the past 70 years, and without these early explorations, this project would not have been possible. That is why it is important to understand both the history of Mars exploration and drone usage to better grasp the significance of this work and what implications it may have for the future.

1.1. A History of Mars Exploration

While several points could be chosen as the origin of Mars exploration, the first truly serious considerations began in the 1950's in the wake of World War II. Wernher von Braun, originally a Nazi rocket scientist who was brought over to the US following the German surrender, became one of the earliest proponents of Mars exploration. From 1952 through 1954 Von Braun published a series of articles in Collier's magazine detailing his vision for space exploration, including a manned mission to Mars. Von Braun's plan outlined a large armada of ships, assembled using an earth orbiting space station, that would launch into Martian orbit and subsequently launch manned gliders to the surface,

specifically the polar ice caps. The crew would then move towards the equator of Mars, construct a new runway which would then be used to land the remainder of the exploration crew. They would then spend over a year surveying Mars's surface before launching back to their original ships and returning to earth (Portree 2001).

More realistic surveys began years later, when NASA began its first study into a Mars expedition in 1960. Early estimates for craft weight and mission duration were made during this initial survey. Researchers, still expecting the mission to be manned, found several obstacles and trade-offs. Shortly before beginning the survey, satellites detected large amounts of radiation in space, raising concerns for astronaut health. Designers had to choose between adding weight for additional shielding to protect the astronauts, or adding fuel to shorten the mission duration, and thus limit the radiation exposure (Portree 2001). They found the more economical approach would be shorter trips with lesser shielding.

The focus on manned missions began to change in the early 1960's as the US and USSR conducted a series of unmanned flybys of Venus and Mars. The first successful flyby of Mars occurred in 1965, when the Mariner 4 managed to obtain 21 pictures of the martian surface and some additional atmospheric data. From this single mission scientists learned that Mars's surface was more arid than previously thought, and had an atmosphere composed mostly of CO_2 with only 1% the density of Earth's (Portree 2001). This was followed by the Mariner 9 mission, which managed to enter Mars's orbit and returned over 2000 images of the planet and its moons (Portree 2001). During this time, the USSR launched its MARS 2 and 3 missions. The former crash landed on the planet, becoming the first man-made object to reach the surface of Mars, while the latter landed successfully, but communication with the probe cut out after 14.5 seconds (Perminov 1999).

These missions were followed by the Viking Landers later in the 70's and the Mars Pathfinder mission in 1997. The Viking Landers were a multiphased expedition, intended as both a broad survey of Mars from orbit, as well as a detailed look at its atmosphere and the possibility of life on the planet

(Corliss 1974). Following a 1975 launch, the probe successfully landed the following year. The lander collected atmospheric data and soil samples from the surface, in addition to the first color images ever of the planet's surface. Meanwhile, the probe's original orbiter was used to map out 97% of the surface of Mars, in preparation for future landings (Portree 2001). This would come in the form of the 1997 Mars Pathfinder mission where, in 1997, NASA landed the first rover on Mars. Sojourner, the rover, collected geological, atmospheric, and rotational data from around its landing site, in addition to high quality images. This data lead to several conclusions, with erosion patterns pointing to liquid water once existing on the surface and rotational data giving a better understanding of the planetary core (Golombek et al. 1997).

Three other rovers have been launched since then, Opportunity, Spirit, and Curiosity, with another expected to launch in July 2020. The Mars 2020 project will see a significant departure from previous missions as it will include a UAV along with its traditional land-based rover. During the mission, the drone will make 5 flights, gradually testing its range and mobility. This design has been in development since 2013 and will serve as a proof of concept for possible future heavier-than-air vehicles to be used on Mars (Northon 2018).

1.2. History of Drone Usage

The idea of unmanned aircraft has existed for almost as long as aircraft themselves have. Although some had toyed with the idea of using radio controls and gyroscopes to guide aircraft remotely beforehand, the advent of World War I sparked the first sizable investment in the technology. British and American researchers, funded by their respective militaries, were both tasked with creating designs for radio controlled planes as a means of flying explosive ordinances into enemy fortifications. However, while several successful tests were performed using these designs, neither the British or American models ever saw combat use before the armistice was signed (Keane and Carr 2013). In modern terms these designs, such as the ‘Kettering Bug’ and ‘Curtiss-Sperry aerial torpedo’, would be considered

missiles instead of UAVs, as they are intended for one time use and expected to be destroyed. This distinction between missiles and UAVs would not become prevalent until World War 2, however (Sullivan 2006).

In the interwar period, the technology would develop further, although at a much slower pace. US Military research into their designs continued until 1925, when they were abandoned due to lack of military interest. British efforts extended past this point and by 1933 they were successfully using radio controlled aircraft, such as the 'Fairey IIIF', as aerial targets in military exercises. After seeing a demonstration of these aircraft, U.S. naval officers were ordered to develop versions of their own. During this development process the Bureau of Aeronautics Lieutenant Commander Delmar S. Fahrney coined the term 'drone' (Keane and Carr 2013). By 1938, the U.S. Navy had begun using UAVs for anti-aircraft target practice (Sullivan 2006).

World War 2 saw renewed efforts to use UAVs for combat purposes, by both the U.S. Navy and Army. A major step forward came as planes were outfitted with television cameras that broadcast footage back to control planes. This allowed for the drones to be precisely controlled from a safe distance of up to six miles (Keane and Carr 2013). The Army's operation, dubbed Project Aphrodite, retrofitted obsolete B-17 bombers with less armor, but heavier payloads, with the purpose of crashing them into German industrial and launch facilities. None of these missions were deemed successful, all failing to properly hit their targets. The Navy conducted a similar operation in Europe, Project Anvil, that ended with similar results (Sullivan 2006). The Navy did have more success in the Pacific theater with its Special Air Task Groups, successfully striking a beached Japanese ship outfitted with anti-aircraft batteries. However, the Navy deemed the project unnecessary as the tide of the war turned in the U.S. favor, and disbanded the unit in 1944 (Keane and Carr 2013).

As the Cold War began in earnest, drone uses started to split between combat purposes and reconnaissance purposes. The first major reconnaissance UAV was the 'Firebee', originally designed in

1951 by the Ryan Aeronautical Company. These were heavily used by the U.S. military during the Vietnam War where they, among other models, flew over 3400 sorties, with only 211 craft ever being lost (Sullivan 2006). These designs were also considered for combat roles, and although they proved more successful than previous models at accurately crashing into enemy targets with payloads, they found greater use in delivering air-to-ground munitions. Development of this concept began in 1971, and by 1973 these new drone models saw use by Israel during the Yom Kippur War, delivering guided munitions against Egyptian targets (Keane and Carr 2013).

Although combat drones were rarely utilized by the U.S. through the 1970's-80's, their successful usage by Israel led to the U.S. developing more sophisticated models which saw use during the Persian Gulf Wars, particularly the Pioneer style UAV. With the Pioneer's overwhelming reconnaissance and logistical success, drones became a standard part of the U.S. military operations, most notably in the modern use of Predator drones in the Middle East (Keane and Carr 2013).

While much attention is paid to the military uses for UAVs, they have also grown increasingly popular for civilian usage since the end of World War 2. The initial market was dominated by hobbyists, primarily flying remote controlled model airplanes. Over the years, new models were developed, such as the extremely popular 'Quad-Copter' and 'Octo-Copter' style of drones (Canis 2015). These, however, remained in the realm of private usage, as FAA rules prevented commercial use of UAVs without express permission until 2015 (*E-CFR: TITLE 14—Aeronautics and Space* n.d.). The effects on the industry were immediate, as revenue from the industry surged from $3.3 billion to $4.5 billion, and is projected to continue growing (Canis 2015; Meola 2017).

With these new rules in place, the possibilities for UAV usage have exploded for both commercial and personal enterprises. Agricultural organizations have discussed using drones to monitor crops and selectively apply pesticides and herbicides. Construction and utilities companies are investigating the use of drones for surveying and inspection purposes. The film industry has already

begun using drones as a cost effective method for getting complicated aerial shots (Canis 2015). Of particular interest to this project are the possible high-altitude usages for drones, as propeller designs for Mars atmosphere will face the same obstacles as propellers designed for high-altitude Earth atmosphere, and could likely be used in both environments.

1.3. History and Principles of Propeller Design

In addition to the examination of drone and Mars exploitation history, a look at both the origins of propeller design and it's fundamental theories is necessary, as this is where the bulk of the design work will be done. Propellers are defined as devices meant to convert power from motors into a forward thrust to power an aircraft, originally called airscrews to differentiate them from marine propellers (Weick 1930; Hitchens 2015). While propellers for the use of windmills have existed in China and Europe for centuries, these were used for converting wind power to axel power, not for any propulsive uses (Hitchens 2015). One of the first airscrew designs with the purpose of powering an aircraft was devised by Leonardo Da Vinci in 1490, although this design was never constructed. The first attempts to use propellers to power flights were performed by early French aviators, such as Jeanne-Pierre Blanchard who unsuccessfully attempted to use a crude airscrew to propel a hot air balloon in 1784. His attempts were followed by Henri Gifford, who in 1852 successfully tested the first powered airship, utilizing a 3 blade propeller. These all paved the way for the Wright brothers, who used a propeller to power the first ever heavier-than-air aircraft at Kittyhawk, NC in 1903 (Hitchens 2015).

With the initial propeller designs came the original math and physics to explain and predict their performance. Among the first and most prominent frameworks was axial momentum theory, sometimes called Rankine-Froude Momentum Theory, after it's major contributors, William Rankine and R.E. Froude (Weick 1930; Hitchens 2015). This theory is a simplified look at how propellers generate thrust, assuming that the propeller acts as an infinitesimally thin actuator disk, with evenly distributed thrust and no

torque. The theory posits that there is an increase in pressure as fluid passes though it to the back end, as well as an additional induced velocity imparted by the disk. This pressure differential between the front and back of the disk is what causes the thrust (Weick 1930; H. (Hermann) Glauert 1948; Gudmundsson 2014). While useful for preliminary estimates of propeller performance, particularly thrust and efficiency, this is a simple theory that ignores the geometry and specific forces at play on the propeller, and tends to return inflated numbers (Gudmundsson 2014).

A more accurate and complete theory of propeller performance was created with Blade Element Theory. This theory framed a propeller blade as being made of several individual airfoil components, each with their own local flow. The individual forces could then be calculated for these elements and summed to find the full force on the blade and propeller. The original idea of analyzing a propeller blade as separate elements came from William Froude, father of R.E. Froude, in 1878. However, most credit Stefan Drzeweicke as the main creator of the theory, as he performed the majority of the research work, created much of the formalism, and brought the theory into general practice. In addition, he also pioneered the practices of summing the forces of the elements and using experimental airfoil data to estimate performance (Froude 1920; Weick 1930; Hitchens 2015). Despite these improvements, this primitive blade element theory proved to be inaccurate (Durand and Lesley 1925; Gudmundsson 2014). This is likely due to it failing to account for the induced velocity presented in momentum theory, as well as ignoring other key concerns, such as tip and hub performance losses (Weick 1930). These effects can be mitigated by adding corrections and taking momentum theory into account, but this will be discussed in further detail later in the paper. Since then, several innovations have occurred, most notably Computational Fluid Dynamics, which combine advanced fluid dynamics math with high processing power to accurately simulate fluid flow around solid objects. These simulations have been found to be accurate and versatile for propellers, even with extreme conditions (Kutty and Rajendran 2017).

With the advent of powered flight and the theoretical physics slowly developing around the field of aeronautics, several different variations on propellers were created. Amongst the earliest and most important propeller designs was Lucian Chauviere's Integrale design, which led him and his company to become the most prominent manufacturers of early propellers (Hitchens 2015). The next major innovation were adjustable propellers, whose pitch could be changed to best suit the flight requirements. These came in several forms, such as the controllable pitch propeller. Perfected by Frank Caldwell in 1933, this design allowed the pilot to adjust the pitch of the propeller in mid flight (Gudmundsson 2014; Hitchens 2015). Another example is the constant speed propeller, patented in 1924, which used a governing control mechanism to balance centripetal and hydraulic forces, and automatically change the pitch to maintain its current RPM (Gudmundsson 2014; Hitchens 2015). Of other great importance was the reverse thrust propellers, operational in 1943, which allowed pitch to be reversed and negative thrust to be generated, useful for slowing down after landing (Gudmundsson 2014; Hitchens 2015). While the advent of turbo props and jet engines would soon dominate the commercial aviation landscape, propellers remain an important component and tool, particularly with regards to drone flight.

1.4. Rovers vs. UAVs

A final but important point that bears discussion is the reasons for using a UAV for Mars exploration and what advantages they may have over rover models. In terms of mobility, UAVs are far superior to rovers, although their lack of durability or power capacity makes them reliant on rovers. The current NASA MARS 2020 project intends to use both (Northon 2018).

Perhaps the biggest advantage that UAVs have over traditional rovers are the greater mobility they provide. Rovers are bulky and heavy by design, ensuring they can reliably carry however much equipment is necessary for their mission. However, their power systems are primarily designed with longevity in mind, not performance. The most recently launched rover, Curiosity, weighed a total of

1,982 lb and was powered by the residual heat of a decaying plutonium dioxide sample, capable of producing 100 W of power (mars.nasa.gov n.d.). For comparison, 2020 Toyota Prius models have weights between 3,000-3,100 lb, but maximum power outputs of 90,000 W ("2020 Toyota Prius Exterior Specs" n.d.). Their low power to weight ratio means that rovers move at extremely slow speeds, with the Curiosity rover having a top speed of 90 m/hr (Greicius 2015). A UAV design, however, is forced to be lightweight due to the low thrust generation in Mars's atmosphere and experience a much higher power to weight ratio as a result. The Mars 2020 UAV has a weight of under 4 lb and is expected to be able to fly 'hundreds of meters' in about 90 seconds (Northon 2018).

Speed is not the only component of this greater mobility. Rovers are stuck moving on the ground, limiting both their field of vision and the types of obstacles it is able to traverse. Curiosity has a maximum height of 7 ft at its 'head' and is only able to roll over obstacles with a maximum height of 29 in (mars.nasa.gov n.d.; Greicius 2015). By contrast, the Mars UAV is expected to be able to hover at 10 ft in the air for 30 seconds, allowing it to easily maneuver around obstacles of that height (Northon 2018). It can also traverse over low and wide hazards such as ravines or gorges, that would be completely inaccessible to a standard rover. These features give the UAV the advantage in both fields of vision and mobility.

These advantages come with costs, and explain why the current Mars 2020 plan is to include both a rover and a UAV. The low weight of UAVs come with several drawbacks that a rover would be well suited to overcome. They are unable to carry the breadth of equipment and tools a rover can, have low energy storage capacity, and have several fragile extremities. A rover can compensate for all of these weaknesses by acting as a 'homebase' for the drone to operate from, while the drone can help the rover by increasing its data and sample collecting capabilities.

CHAPTER 2

SIMULATING THE PROPELLER

The goal of this project is to design a propeller that can function in the Martian atmosphere. There are general principles of good propeller design that can be followed, regarding thrust output and efficiency, but these are simple guidelines, and useless without a way of estimating how certain design choices may affect the performance in this environment (Gudmundsson 2014). As such, this project requires a method of simulating propeller performance given various design parameters.

The chosen method for this task was Blade Element Momentum Theorem (BEMT), a simple, but efficient way of estimating the thrust, torque, and power of a spinning propeller. This method allows for rapid design testing and can be modified and corrected to account for specific issues with the testing environment. This has proven to be a simple, and relatively accurate method to gauge propeller performance, and has great potential to be refined for higher accuracy (Gudmundsson 2014).

2.1. The Basics of Blade Element Momentum Theory

The key values calculated by BEMT are thrust, torque, and power of the propeller, all of which are based on the total lift and drag values of the individual blades. The usual calculations for lift and drag are integrals taken over the entire blade.

$$L = \int_{R_{HUB}}^{R} \tfrac{1}{2} \cdot \rho \cdot V_E^2 \cdot c \cdot C_l \cdot dr \qquad (2.1)$$

$$D = \int_{R_{HUB}}^{R} \tfrac{1}{2} \cdot \rho \cdot V_E^2 \cdot c \cdot C_d \cdot dr \qquad (2.2)$$

In these equations R and R_{HUB} are the total radius and hub radius of the propeller, ρ is the atmospheric density, V_E is the effective resultant velocity, c is the chord length, C_l and C_d are the lift and drag coefficients, and dr is the radius differential. There are several issues with using these exact equations for propeller design. The first is that V_E, c, C_l, and C_d are all dependent on the radial position and thus need to be replaced with larger, more complex formulas, complicating the integral. V_E in

particular requires a lengthy algorithm to account for induced velocity effects of the propeller stream tube, and this cannot be truly captured by the integrals (Gudmundsson 2014). Corrections to other properties such as C_l and C_d also suffer from this issue. Further, these only yield a final overall number for lift and drag, and do not allow the designer to easily assess how performance at different radial positions may compare to one another.

These complications are what make BEMT so appealing. Instead of taking one large integral over the entirety of a propeller blade, BEMT divides the blade into individual blade sections, treats them as nearly 2-D airfoils, calculates all of their individual lifts and drags, and adds them all together to give an estimate for total drag and lift of the blade (MacNeill and Verstraete 2017). This is similar to how Simpson's Rule breaks integrals into individual components and adds their areas to find the overall integral. In addition to the simplicity of the calculations, BEMT can also return performance values for individual blade sections, and thus allow the designer to better examine areas for improvement for the propeller.

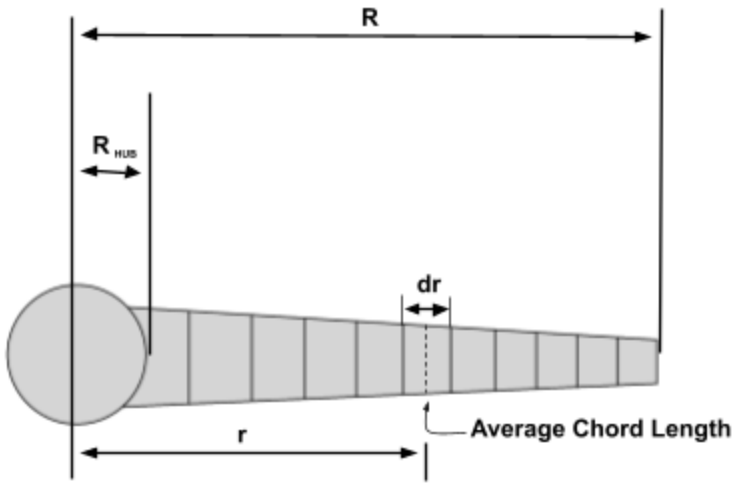

Figure 2.1: Propeller Blade Division using BEMT, derived from General Aviation Aircraft Design (Gudmundsson 2014)

Before any of these calculations can take place, both design and atmospheric parameters must be specified. The Design section of this paper will look into the specific design choices made for these propellers, but for this particular formulation the radius, hub radius, chord length, number of blades, airfoil, twist, RPM, and forward airspeed must all be known before the calculation can take place. In addition to these design specifications, the atmosphere that is being simulated must also be given. The values needed for BEMT are pressure, density, temperature, and viscosity. For this project's simulation, sample atmospheric data from Mars was used, taken from Solid State Aircraft Phase II Project NAS5-03110 Final Report, with official numbers recorded in Appendix A (Colozza et al. 2005). It should be noted that while there are more general models for the Mars atmosphere, this data set was chosen as it better explores the altitudes in which a UAV on Mars may operate.

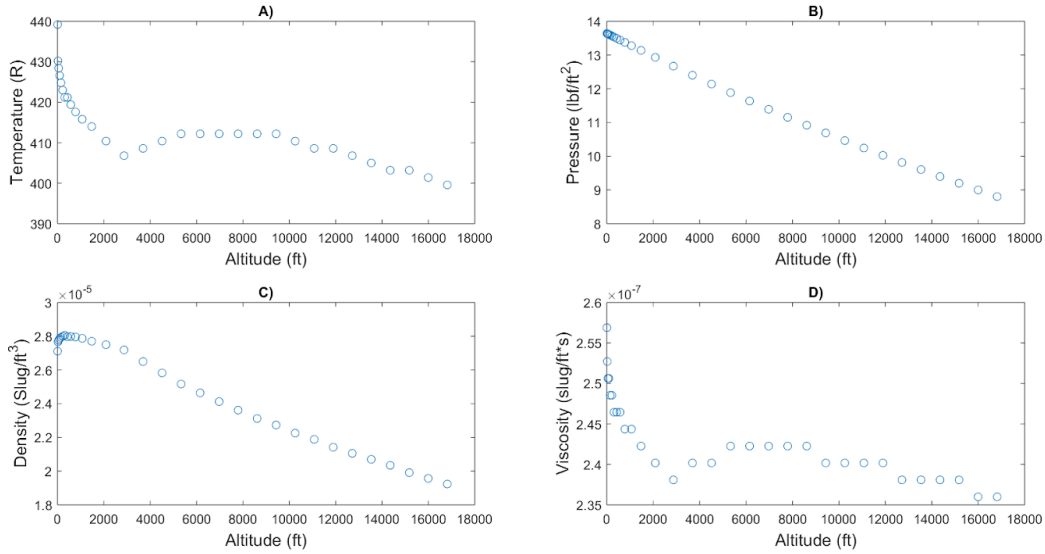

Figure 2.2: Atmospheric Data from JPL Reference Mars Atmosphere for -20° Latitude Data A)Temperature B)Pressure C) Density D)Viscosity

From this data, linear best fit lines were generated, using altitude as the independent variable. Although these graphs are not all specifically linear, this is data from a single point in the atmosphere.

Earth's atmosphere can be modeled using linear equations at altitudes below 11,000 m, and because this UAV is not intended for high altitudes on Mars, linear equations should suffice here (Anderson 2012).

$$Temperature = 422 - (0.00131 \cdot h) \quad (2.3)$$

$$Pressure = 13.6 - (0.000294 \cdot h) \quad (2.4)$$

$$\rho = 0.0000282 - (5.52 \bullet 10^{-10} \cdot h) \quad (2.5)$$

$$\mu = 2.47 \bullet 10^{-7} - (6.98 \bullet 10^{-13} \cdot h) \quad (2.6)$$

This allows the designer to enter a single altitude, and for the algorithm to then give an accurate estimate of what the atmospheric conditions may be. All of these values are used in the BEMT algorithm, with the exception of pressure. Pressure will be used during the CFD simulation, however, so having an established model is necessary.

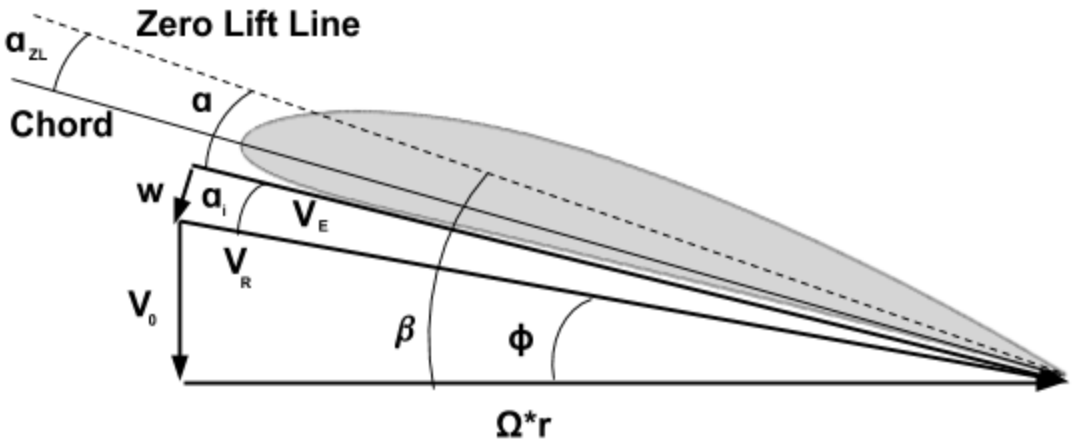

Figure 2.3: Angle and Velocity Diagram of a Blade Section, using set up from General Aviation Aircraft Design (Gudmundsson 2014)

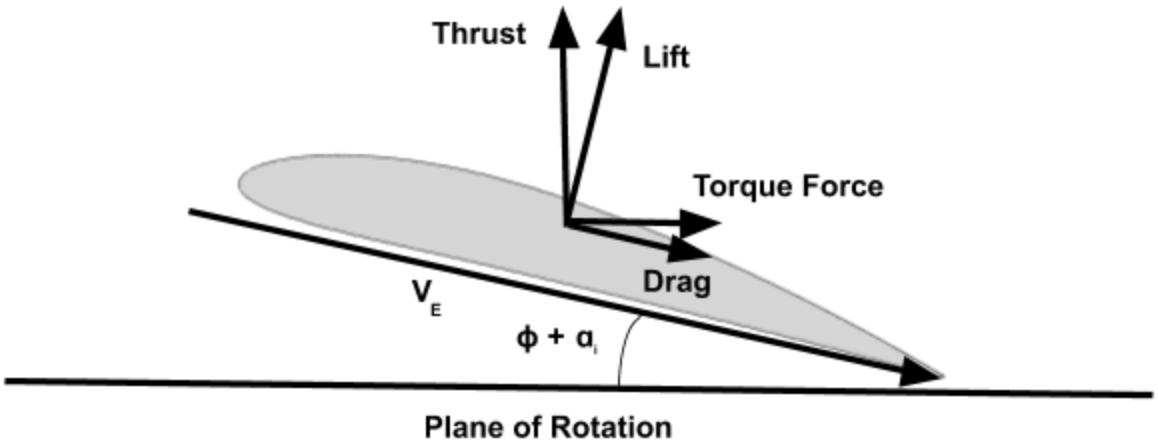

Figure 2.4: Forces Diagram of a Blade Section, using set up from General Aviation Aircraft Design (Gudmundsson 2014)

With the atmosphere set, the algorithm can begin. The first phase takes the blade, and breaks it into equally spaced sections along its radial dimension. The number of stations can be altered to the designer's preference, with higher numbers providing more accurate estimates. For this simulation, 500 blade stations were used. After the stations are generated, their chord length and twist angle can be calculated, based on the inputs of the designer. In addition to this, their rotational speed is calculated using the given RPM and distance from the center of the rotor. This can then be combined with the forward velocity, using vector addition, to obtain the relative airspeed and helix angle of the individual blade stations. This provides the algorithms first estimate for the effective airspeed of each blade station, which will be refined with a correction algorithm accounting for the induced airspeed of the propellers stream tube.

Induced velocity and angle of attack (AOA) are both derived from Rankine-Froude Momentum Theory, which treats propellers as infinitesimally thin disks that impart momentum and pressure change on the fluid passing through their front facing area, called the stream tube (Gudmundsson 2014). These changes in momentum and pressure that the propeller imparts create a different flow immediately in front of and behind the propeller, separate from the free flow occurring outside of the stream tube. This

new flow alters the effective velocity around the propeller and subsequently the AOA at which this velocity occurs. As such, it requires a correction to both, which are called the induced velocity and induced AOA (Gudmundsson 2014).

The importance of these induced values is found when considering which other values in BEMT rely on accurate velocity and AOA. Lift and drag are both quadratically proportional to V_E, but the dependence goes further than this. Both the C_l and C_d values are dependent on AOA and Reynolds number (Re) of the flow, which itself is dependent on V_E. As such, failing to account for these induced values can lead to derived values being inaccurate.

In order to accurately estimate the induced velocity and AOA of the stream tube, a new algorithm based on Momentum Theory must be used. Momentum Theory provides a way of estimating these values; however, it requires V_E, C_l, and C_d which means that once a new velocity and AOA are calculated, these initial values then become inaccurate. Thus there needs to be an iterative method for deriving these induced values, where they are constantly recalculated until they begin to converge on a single value. The algorithm adopted in this instance is a form of the Newton–Raphson Method, with a simplified function derived from momentum theory taken directly from General Aviation Aircraft Design (Gudmundsson 2014).

$$f(w) = \frac{8\pi r}{N_B \cdot c} \cdot w - \frac{V_E}{(V_0 + w)} \cdot [C_l(\Omega r) - C_d(w + V_0)] \quad (2.7)$$

$$f'(w) = \frac{8\pi r}{N_B \cdot c} - C_l(\Omega r) \cdot \left(\frac{1}{V_E} - \frac{V_E}{(V_0 + w)^2}\right) + C_d \frac{(w + V_0)}{V_E} \quad (2.8)$$

$$w_{n+1} = w_n - \frac{f(w)}{f'(w)} \quad (2.9)$$

$$w_{dif} = w_{n+1} - w_n \quad (2.10)$$

This process is then repeated until the $w_{dif} < 0.0001$, indicating the value of w has converged sufficiently. With each iteration of this calculation new values for Re, AOA, C_l, and C_d are calculated in order to keep the estimates accurate.

Both C_l and C_d are necessary to calculate propeller performance and vary with Re and AOA, meaning there needs to be a function for both coefficients dependent on these values. XFLR5, a free program using XFOIL code, can be used to calculate these coefficients over several Reynolds numbers and a range of AOA. Mach number is always assumed to be 0, as it is corrected later in the BEMT program, and N_{CRIT} is kept at the default value of 9 unless otherwise needed. The ranges of both Re and AOA vary depending on the particular airfoil and situation being examined, with other variations occurring based on XFLR5's ability to generate acceptable values. These deficiencies are accounted for with later corrections (see section 2.5). Next the raw polar data (lift and drag values as a function of AOA and Re) is imported into a spreadsheet program and best fit curves are attached to them. These are done individually for C_l and C_d at a given Reynolds number, with AOA being the independent variable and the coefficient being dependent. Best fit lines generally take the form of 4th-6th order polynomials, depending on the shape of the polar. These equations are then placed into the final BEMT program, where the coefficient values can be calculated at the set Reynolds numbers, based on the AOA of the propeller blade stations. As the blade stations don't fall on even intervals of Reynolds numbers, the final coefficient values are linearly interpolated from set Re values. Cases where Re < 1000 are interpolated with values of 0. Although this may not be entirely accurate, Re < 1000 typically only occurs at low speed areas near the hub. The low speed and proximity to the hub significantly lower the performance of this section, and as such values here have much smaller effects on the overall propeller performance. This method of calculating values at several Reynolds numbers and interpolating between them is recommended by several sources, and found to give more accurate final results (Gudmundsson 2014; MacNeill and Verstraete 2017).

Once this process is complete, and accurate values for V_E, C_l, and C_d are obtained, the blade element values of lift and drag can be determined. These simply use the derivative form of equations (2.1) and (2.2), with dr being the width of each element.

$$dL = \tfrac{1}{2} \cdot \rho \cdot V_E^2 \cdot c \cdot C_l \cdot dr \qquad (2.11)$$

$$dD = \tfrac{1}{2} \cdot \rho \cdot V_E^2 \cdot c \cdot C_d \cdot dr \qquad (2.12)$$

From these sectional values, the radial distance of each blade element, the helix angle of each element, the induced AOA of each element, and the rotational velocity, the sectional values of thrust, power and torque can be determined (Gudmundsson 2014).

$$dT = dLcos(\varphi + \alpha_i) - dDsin(\varphi + \alpha_i) \quad (2.13)$$

$$dQ = r \cdot [dLsin(\varphi + \alpha_i) + dDcos(\varphi + \alpha_i)] \quad (2.14)$$

$$dP = \Omega r \cdot [dLsin(\varphi + \alpha_i) + dDcos(\varphi + \alpha_i)] \quad (2.15)$$

After these sectional values are found, they can be summed and multiplied by the number of blades to find the thrust, torque, and power of the total propeller. This is the simplest form of BEMT, and ignores many of the further corrections that need to be made to account for the nature of fluid flow.

2.2. Model Corrections

The BEMT algorithm described above can be used to obtain a fairly accurate estimate of propeller performance, but further modifications can be made to ensure more accurate results. Currently the algorithm doesn't account for losses in performance near the edges of the blade, compressibility effects, or general effects due to rotation. These corrections will further refine the estimate, creating numbers which more accurately depict reality.

One necessary correction is fixing how lift and drag are calculated around the tip and hub of the propeller. At the tips of finite blades, there is a cross flow of fluid that decreases the pressure difference between the lower and upper part of the blade, leading to a loss of lift and torque (Vries 1979). A similar issue occurs at the hub of the propeller as well. As such, this requires a correction for these performance losses, proportional to the distance from the hub or the tip. Ludwig Prandtl determined a method for doing so which easily fits into the BEMT framework. While experiments have shown this correction to be accurate, the accuracy drops for propellers with less than three blades and high tip speed ratios, both

typically attributes of UAV propellers. As such a small correction factor can be added to further raise the accuracy of the correction (Masters et al. 2011).

$$g = e^{-[c_1 \cdot (N\lambda - c_2)]} \quad (2.16)$$

$$P_{Tip} = g \cdot \frac{N_B}{2} \cdot \frac{(R - r)}{r \sin(\varphi)} \quad (2.17)$$

$$F_{Tip} = \frac{2}{\pi} \cdot \cos^{-1}(e^{-P_{Tip}}) \quad (2.18)$$

$$P_{Hub} = g \cdot \frac{N_B}{2} \cdot \frac{(r - R_{Hub})}{r \sin(\varphi)} \quad (2.19)$$

$$F_{Hub} = \frac{2}{\pi} \cdot \cos^{-1}(e^{-P_{Hub}}) \quad (2.20)$$

$$F_P = F_{Tip} \cdot F_{Hub} \quad (2.21)$$

In these equations, g is a correction factor based on blade number and tip speed ratio, where c_1 and c_2 are set to 0.125 and 21, respectively. These numbers were determined experimentally for this correction (Masters et al. 2011). P_{Tip} and P_{Hub} are both correction vectors based on proximity to either the total radius or hub radius. F_{Tip} and F_{Hub} are the actual correction factors for the lift and drag of the blade, and F_p is the final product of the two (Gudmundsson 2014). The sectional thrust and torque values are then multiplied by F_p, with the strongest corrections occurring at the tip or the hub, where the correction values were lowest, while the middle of the blade remains relatively unaffected.

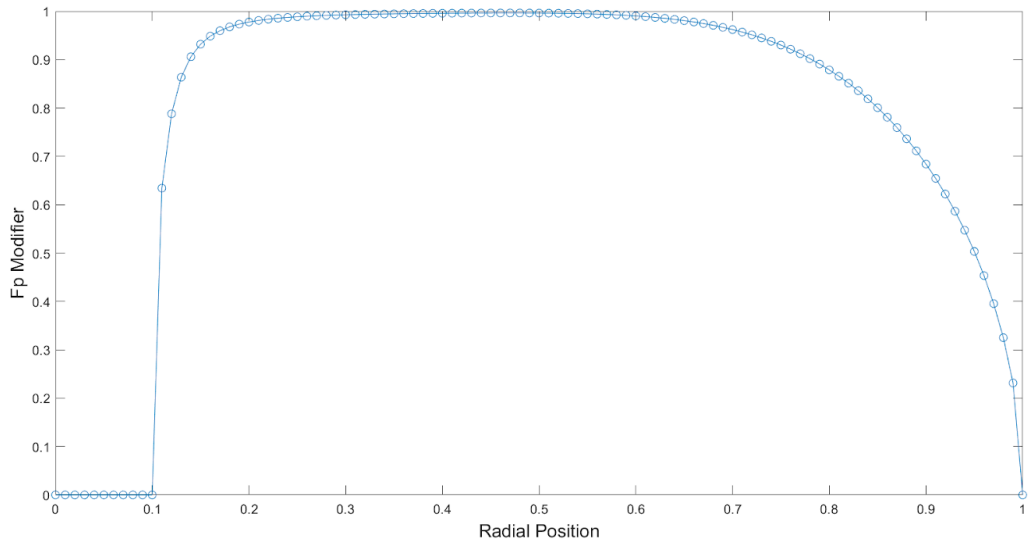

Figure 2.5: Chart showing Fp modifier for a blade with low speed and AOA. Note the total loss of performance at the hub, and the gradual loss near the tip

Another correction that needs to be taken into account is the effect of compressibility on the lift and drag performance. Regular BEMT calculations assume that the fluid is incompressible, which is naturally untrue, particularly in low density gas environments. As the propeller spins, the surrounding air will be compressed in proportion to the Mach Number of each blade section (H. Glauert 1928). This translates to stronger effects near the tip of the blade (Gudmundsson 2014). Specifically, the rise in fluid density around the propeller will lead to an overall rise in both the lift and drag. This effect is especially important for a propeller designed for the martian atmosphere, as high rotational velocities and Mach Numbers will be necessary to overcome the lack of density in the atmosphere. The Prandtl-Glauert correction method was utilized for correcting the lift coefficient, as it is easily compatible with BEMT and is accurate for the Mach number regime used in this project (H. Glauert 1928).

$$M = \frac{V_E}{a} \quad (2.22)$$

$$C_l = \frac{C_{l0}}{\sqrt{1-M^2}} \quad (2.23)$$

In these equations M is the mach number, a is the speed of sound, derived from the temperature and composition of the atmosphere, and C_{l0} is the uncorrected lift coefficient. As the Mach Number rises, C_l will increase in proportion. It should be noted, however, that as Mach numbers reach regimes of M = 0.7-0.75, this correction becomes inaccurate (Gudmundsson 2014). To account for this, the simulation code has an if statement checking for the Mach Number of each blade station. If it is in excess of 0.7, then the correction is dropped, and the simulation is given an indicator that it exceeded the Mach Number limit.

A separate approach is needed for drag. Drag is typically divided into two categories, pressure and skin friction, and thus the correction needs to be more nuanced. The Prandtl-Glauert method works for purely pressure based coefficients, such as lift and moment, and thus cannot be used for the multifaceted drag coefficient (Gudmundsson 2014). What can be used, however, is the Frankl-Voishel, which applies a mach number based correction to the skin friction drag (Gudmundsson 2014). This can then be added to the pressure drag for a final corrected drag value. This method was originally used in the program, but was later abandoned due to other corrections for drag being inapplicable to pressure drag and skin friction drag individually, leaving this correction inaccurate. However, due to skin frictions relatively low value compared to overall drag, this likely has minimal effect on final values.

$$C_{Df} = C_{Df0} \cdot (0.000162M^5 - 0.00383M^4 + 0.0332M^3 - 0.118M^2 + 0.0204M + 0.996) \quad (2.24)$$

A final necessary correction for the simulation is accounting for the rotational effects of the spinning propeller. Through both theoretical and experimental work it has been found that the rotation of propeller blades has a delaying effect on laminar separation, which in turn raises both the C_{lMAX} value as well as the stall angle of individual airfoil sections, with effects being more pronounced towards the hub rather than the tip (Corrigan and Schillings 1994; Lindenburg 2003). This alters the shape of the lift curve for airfoils, and will lead to underpredicting performance when estimating with exclusively 2-D

airfoil data. As such, a method needs to be utilized to both raise the C_{lMAX} as well as pushing back the stall angle.

Several methods have been developed to accomplish this goal with varying levels of accuracy. For this simulation, the method developed by Corrigan and Schillings was selected for its ease in integrating with BEMT and XFOIL data, and also its accuracy when compared to other methods (MacNeill and Verstraete 2017; Tangler and Selig 1997; Corrigan and Schillings 1994). This method is based on chord to radius ratios, allowing for effects to change from hub to tip, where it pushes the C_{lMAX} angle back before correcting the lift coefficient based on the new slope of the curve.

$$K = \left(\frac{0.1517}{c/r}\right)^{0.9225} \quad (2.25)$$

$$\Delta\alpha = (\alpha_{Clmax} - \alpha_{Cl=0}) \cdot \left[\left(\frac{K \cdot (c/r)}{0.136}\right)^{n} - 1\right] \quad (2.26)$$

$$C_{lRot}(\alpha + \Delta\alpha) = C_{lNon-Rot}(\alpha + \Delta\alpha) + \left(\frac{\partial C_l}{\partial \alpha}\right) \cdot \Delta\alpha \quad (2.27)$$

Here c/r is the chord to local radius ratio, C_{lRot} is the corrected lift coefficient for rotation, and n is an exponent, with values between 0.8 and 1.6 giving accurate results (MacNeill and Verstraete 2017). In this instance a value of 1 was used, in line with previous uses of this correction (Tangler and Selig 1997; MacNeill and Verstraete 2017).

2.3. Flow Conditions in Low Reynolds Number Environments

An important aspect of properly simulating rotation is ensuring that flow is accurately being simulated at the required Reynolds numbers. Due to the low-density of Mars's atmosphere, the operating regime of the propeller designs is generally Re < 20,000. This can naturally change with size and RPM adjustments, but no designs examined in this experiment ever exceeded this amount. Re < 20,000 is an extremely low Reynolds number regime in which to operate. As such, there is a lack of experimental airfoil data and theoretical analysis in this regime.

There are some general trends that can be observed at these extremely low Reynolds numbers, however. The most noticeable is the dramatic reduction in the C_l/C_d ratio. At Reynolds numbers

exceeding 100,000 it is common to find C_l/C_d to be on the order of magnitude of 10 or 100. However, as Reynolds numbers shrinks to the range of Re < 20,000, this ratio also drops by a significant amount. Figure 2.3 shows this trend, using a NACA 4412 airfoil as an example. Due to thrust being heavily dependent on lift, and power being heavily dependent on drag, this ratio change results in a significant loss of efficiency for propellers operating within this regime.

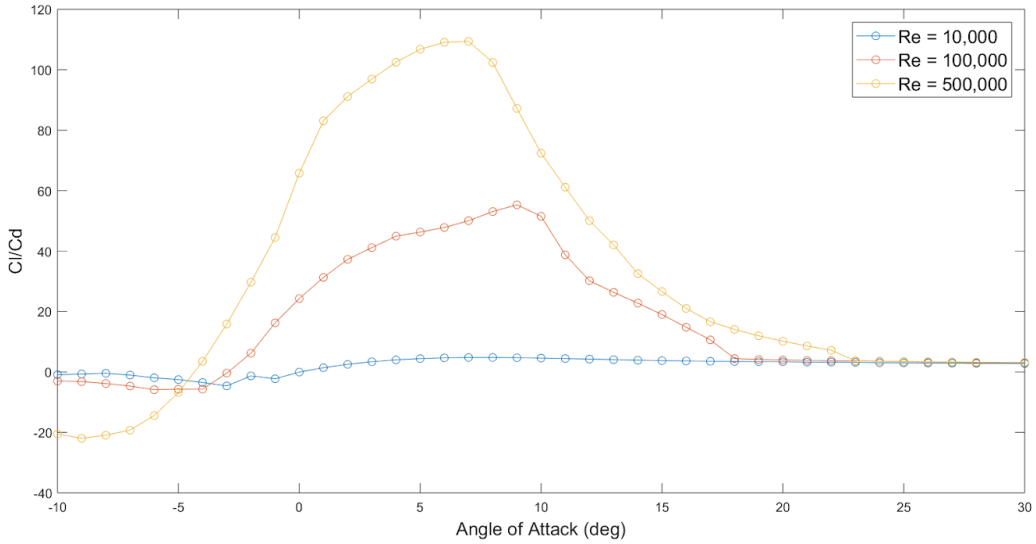

Figure 2.6: Graph Comparing C_l/C_d for NACA 4412 Airfoil, using XFOIL Code

In addition to this loss in efficiency, there are also effects related to the odd 'boundary layer' found in low Reynolds number flow. Because flow operates differently in this regime, boundary layer is more defined as an area close to the surface of the where pressure is relatively stable, as opposed to the typical definition involving laminar and turbulent flow (Kunz 2003). This is a result of viscosity dominating flow at this regime, creating a thick boundary layer, and distorting the geometry of the airfoil. A consequence of this is, as Reynolds number drops, both the minimum coefficient of pressure experienced by the airfoil as well as the slope of its pressure recovery fall, affecting both the lift and pressure drag coefficients. This effect is similar to what happens during flow separation at higher Reynolds numbers, except in this instance the flow is completely attached (Kunz 2003).

The reduced pressure recovery slope leads to other effects as well. One of particular interest is that it delays the onset of separation and stall, allowing for higher angles of attack and higher C_l values while in steady state. Once flow begins to separate, lower Reynolds numbers also delay the separation bubble as it moves up the chord with higher angles of attack. This only reinforces the previous notion, further pushing back the stall angle and raising the C_{lMax} (Kunz 2003).

There are no specific corrections for these phenomena, but they are important to note. With Re < 20,000 being a relatively unexplored regime, using coefficients from flow solvers such as XFOIL bear some uncertainty, without experimental values to confirm them. As such, if the consequences of low Reynolds number flow are observed, such as high stall angles, high C_{lmax}, and low C_l/C_d then more confidence can be placed in the generated values. These observations also give a better understanding of flow in this regime, and could lead to further corrections.

2.4. The Accuracy of XFOIL

All lift and drag coefficients for this simulation are derived from an initial calculation using XFLR5, a free to use analysis tool, utilizing XFOIL code. With so much of the simulation relying on the numbers generated by this program, it is important to understand its strengths, weaknesses, and the overall accuracy of the software. From here, the data can be used or corrected as needed.

XFOIL was designed for the purpose of rapid airfoil analysis and revision, with a particular focus on low Reynolds number (Re < 500,000) regimes. This means an emphasis on airfoils with a transitional separation bubble, and utilizing methods with inverse solutions for inverse design methods. An inviscid linear-vorticity panel method is paired with a two-equation lagged dissipation integral method for viscous layers, with some additional compression corrections, and both are solved simultaneously using a global Newton Method (Drela 1989). This method was specifically designed for rapid prototyping, and quick revisions of airfoils using inverse design methods.

A major problem with testing the accuracy of XFOIL for this project is the lack of experimental airfoil polars at the necessary Reynolds numbers or at the high angles of attack this design will likely utilize. As such, much of the data used for accuracy checks will be at higher Re values and lower AoA values. This makes understanding the effects of extremely low Reynolds numbers detailed in the above section important, and informed estimates and corrections may be necessary.

It is also important to define what 'accuracy' means in this instance. There are many factors and properties to airfoil polars and it is more than possible for XFOIL to accurately match some aspects while missing on others. A few key aspects looked at for the purposes of this project are the shape of the lift curve, the values of C_l and C_d, how they change with regards to Reynolds number and Angle of Attack, where stall occurs, their role in the BEMT model, and what C_{lMax} value it experiences at this point.

One study looks at BC 2125, BC 3111, BC 3X92, and BC S127 airfoils, experimentally testing their accuracy over a Reynolds number range of 60,000-250,000 and an AOA range of -5^o to $+20^o$ (Chen and Bernal 2008). Here several general trends emerge. The most immediately noticeable is that on all airfoils measured at Re = 60,000, XFOIL consistently overpredicts C_l values during pre-stall, has a lower stall angle, predicts higher C_{lmax} values, exaggerates the drop in performance that occurs at stall, and also overpredicts C_l along higher AoA portions of the recovery curve. When comparing results for these airfoils at higher Reynolds numbers, XFOIL is shown to more closely resemble the shapes and values of the C_l curve, although there remains some discrepancy with over prediction and mis-placing the stall angle (Chen and Bernal 2008). These results point to the possibility of XFOIL becoming less accurate at lower Reynolds numbers. The paper also notes that XFOIL frequently predicts the wrong size and location of laminar separation bubbles (LSB), which could pose issues with the highly viscous flow considered in this project (Chen and Bernal 2008).

In another study of the accuracy of XFOIL, similar results were found. This study compares experimental polars with various flow prediction software including XFOIL (Maughmer and Coder 2010).

The range of Reynolds numbers used in this study is large with Re = 70,000-1,500,000, with AOA having a similar range to the previous study in most instances. The results mimic the first study, with C_l and C_{lMAX} generally being overpredicted over the whole range of Reynolds numbers, with a maximum over prediction of 15% for C_{lMAX}. It is noted that C_d is typically very accurate at most of the Reynolds numbers measured, being attributed to the boundary layer methods utilized by XFOIL. Most importantly for this project, they observed that XFOIL largely overpredicts C_l for the S407 airfoil at Re = 70,000, and fails to converge before ever reaching stall. This again casts doubt on the accuracy of XFOIL data at low Reynolds numbers (Maughmer and Coder 2010).

There is another key conclusion from this study worth discussing. It explicitly notes that the programs utilizing integral boundary-layer methods, such as XFOIL, predict C_{lMAX} best when there is a steep pressure recovery gradient on the upper surface of the airfoil, allowing the tail end separation bubble to move up the chord length quickly (Maughmer and Coder 2010). This is the exact opposite of what occurs in this project's Reynolds number regime, where relatively gradual pressure recovery gradients cause separation bubbles to move slowly. As such, it is expected that XFOIL will overpredict C_l values for foils designed for this experiment, and thus could give larger than expected results.

These results were all of airfoils taken at a range of Re=60,000-1,500,000 and AOA = -5^o to $+20^o$. The design range for this project is mostly in the realm of Re < 20,000 and AOA = 0^o to $+30^o$. As such other results should be looked at that better fit the design range of this project. No data could be found for airfoils at very high AOA, but data does exist for several airfoils in the range of Re = 17,000-60,000 (Miley 1982). As such, a number of these airfoils were taken and their polars were recreated in XFLR5 to match their wind tunnel test conditions. All Mach numbers were set to 0, as it can be assumed at these low Reynolds numbers airspeed was likely also low. A number of the experimental results were taken from earlier experiments in the 1950's where turbulence was a larger issue in wind tunnels. To account

for this, these runs all had their N_{CRIT} set to a new value based on the wind tunnels turbulence (Shi et al. 2018).

$$N_{Crit} = -8.48 - 2.4 \cdot ln(\frac{Tu(\%)}{100}) \quad (2.28)$$

For these older tests, N_{CRIT} was set to 5.5, while newer tests were kept at their default value of 9. This default value is closer to the turbulence experienced in modern wind tunnels. Although most wind tunnel test turbulences give a slightly different N_{CRIT} value around 8.6, it has been found that changing this value by small amounts does little to improve accuracy (Chen and Bernal 2008). XFOIL and experimental results were compared for NACA009, E61, GOE 795, GOE 796, GOE 797, and GOE 801 airfoils at a range of Re = 17,000-60,000 and an AOA range of -5°-21°. A selection of these graphs at the lower end of the Reynolds number range are shown in Figures 2-5.

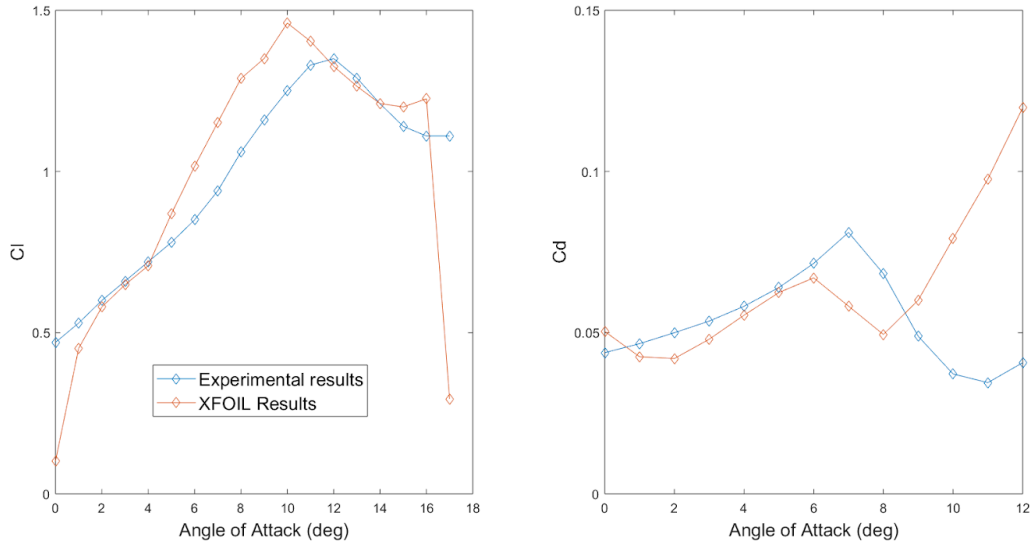

Figure 2.7: Graph Comparing XFOIL and Experimental results for C_l and C_d for the E61 airfoil at Re = 40,000 (Miley 1982)

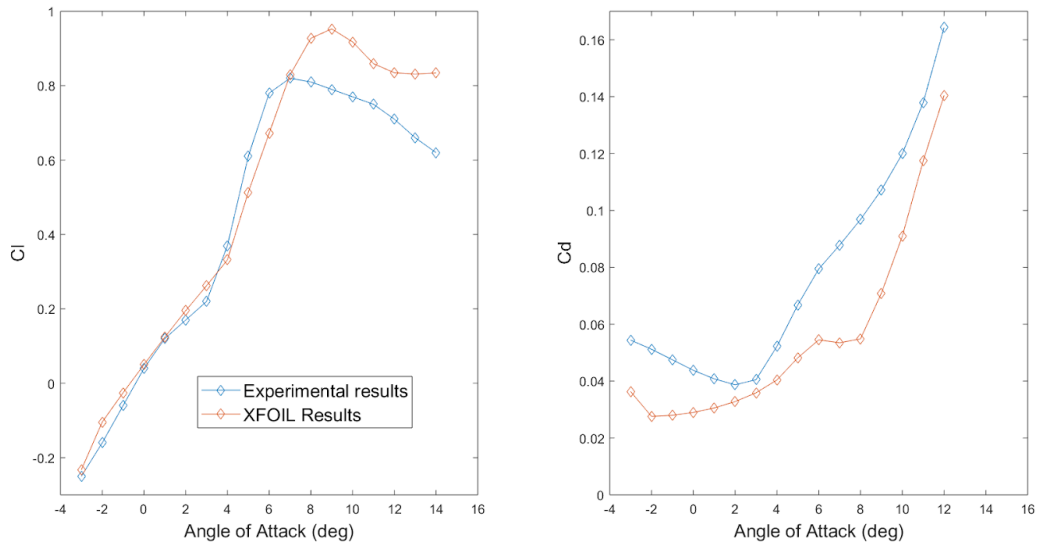

Figure 2.8: Graph Comparing XFOIL and Experimental results for C_l and C_d for the GOE795 airfoil at Re = 17,000 (Miley 1982)

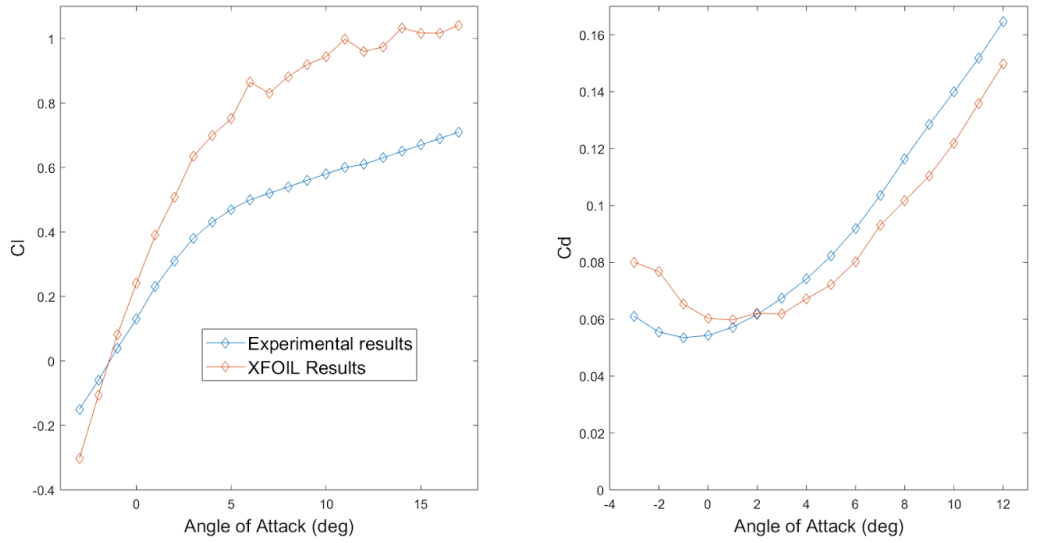

Figure 2.9: Graph Comparing XFOIL and Experimental results for C_l and C_d for the GOE801 airfoil at Re = 21,000 (Miley 1982)

Here, we see results similar to what was seen in the previous studies. While C_l can remain fairly close during pre-stall, it begins to separate around the stall angle, frequently overpredicting C_l. This

varies based on airfoil, with some pushing stall either forwards or backwards, and certain polars significantly overpredicting the drop which occurs at stall. These graphs also show a general tendency to underpredict C_d, although again, this varies from airfoil to airfoil.

These inaccuracies in the aeronautics coefficients will lead to the final values given by the BEMT program to also be inaccurate. As such, steps must be taken in order to correct these coefficients and give a more realistic value.

2.5. Methods of Coefficient Correction

A method for correcting lift and drag coefficients is required to give accurate final performance results for propeller designs. Various methods have been developed for doing so, although they vary in approach, AOA and Re ranges over which they work, and overall accuracy. The strengths and weaknesses of these methods were evaluated against the parameters of the experiment before a final selection was made.

One such method was developed by NASA in 1982, specifically for rotor blades. They found issue theoretical calculations relying on data sets that could only be found for one Reynolds number and Mach number at a time, creating incremental tables of data to use, and leaving interpolation as the only option for finding in between values. This was found insufficient with regards to Reynolds number scaling, and a new method was developed based on experimental data, with a particular focus on C_{lMAX} and C_{dMIN} values. An exponential scaling factor dependent on Reynolds number was found (Yamauchi 1983).

$$K = \left(\frac{Re_{Table}}{Re_{Local}}\right)^{-n} \quad (2.29)$$

$$\bar{\alpha} = \alpha_{zi} + \frac{(\alpha - \alpha_{zi})}{K} \quad (2.30)$$

$$C_{d,\,Local}(\alpha) = \frac{C_{d,\,Local}(\alpha)}{K} \quad (2.31)$$

$$C_{l,\,Local}(\alpha) = K \cdot C_{l,\,Table}(\bar{\alpha}) \quad (2.32)$$

Where n is a value between 0.2 - 0.5. While this algorithm may provide better scaling based on Re, it was not adopted. This method relies on previous experimental results in order to give accurate coefficients. Because this design will be using several custom airfoils, there is no previous experimental data for them. This means the correction would be forced to rely on XFOIL data in lieu of the experimental data, which has been established to be inaccurate. Because this method relies on XFOIL data, but has no means of correcting for its given values, it cannot be relied on for this project.

Another method for correcting coefficients is the Viterna-Corrigan method, specifically designed for low speed wind turbines. This method is based on an idealized version of lift and drag polars where lift experiences a post stall recovery in performance and drag slowly rises to a maximum value at AOA = 90°. It has been found to be very accurate, particularly in the post-stall range of AOA (Viterna and Corrigan 1982; MacNeill and Verstraete 2017). It ignores the typical, brief drop in lift after stall, although this seems to have little effect on the final values of BEMT (Tangler and Ostowari 1991). There are also several methods used for calculating the maximum drag, all of which are based on the AR and sometimes thickness of the blade. For this program, the method yielding the largest C_{dMAX} was chosen, given the effects of extremely low Re on drag.

$$B_1 = C_{dMAX} = \frac{1 + 0.0065AR}{0.9 + (t/c)} \quad (2.33)$$

$$A_1 = \frac{B_1}{2} \quad (2.34)$$

$$A_2 = C_{ls} - [B_1 sin(\alpha_s) cos(\alpha_s)] \quad (2.35)$$

$$B_2 = C_{ds} - \frac{B_1 sin^2(\alpha_s)}{cos(\alpha_s)} \quad (2.36)$$

$$C_l = A_1 sin(2\alpha) + A_2 \frac{cos^2(\alpha)}{sin(\alpha)} \quad (2.37)$$

$$C_d = B_1 sin^2(\alpha) + B_2 cos(\alpha) \quad (2.38)$$

There are a few caveats when utilizing this method. Most of its accuracy is relegated to the post stall range of AOA, making it unhelpful with regards to lower AOA. It also utilizes both C_l and C_d values at

stall, to keep continuity with the pre-stall curves, meaning it is also reliant on XFOIL data to determine when stall is and what the coefficient values are at this point. As such, this method is an incomplete correction.

With most correction methods reliant on the XFOIL data for their calculations, it is necessary to have a method for correcting this specific data to better reflect reality. As no explicit method seems to exist for the range of this project, one was created. The low Re data taken from Miley, 1982 and its equivalent XFOIL data were imported into MATLAB with the purpose of using nonlinear regression to create a model for correcting the coefficients (Miley 1982). Other data such as thickness, camber, and Reynolds number were also taken, and can be viewed in Appendix A.

To create a model specific to input data, MATLAB requires an initial model with unknown β coefficients. It then takes the data given and a vector of initial coefficient values, assigns it to the specified variables in the model, and iteratively alters the coefficients to minimize the mean square differences between the final model values and a vector of desired values ("Nonlinear Regression - MATLAB & Simulink" n.d.). In the case of this model, the desired values are the experimental coefficients.

The goal of this model was to take the XFOIL data and correct it to the experimental values, dependent on the AOA, Re, t/c, and camber. As there were no previous models done with this method, several versions were attempted. The first method raised thickness, camber, and Re to various exponents, while treating AOA as a fourth order polynomial series.

$$F(\alpha) = \beta_1 + \beta_2\alpha + \beta_3\alpha^2 + \beta_4\alpha^3 + \beta_5\alpha^4 \quad (2.39)$$

$$C_{l,Corrected} = \beta_6 \cdot Re^{\beta_7} \cdot (t/c)^{\beta_8} \cdot (Camber\%)^{\beta_9} \cdot F(\alpha) \cdot C_{l,XFOIL} \quad (2.40)$$

This method was selected based on the apparent patterns observed in the polar data. Over and under prediction could vary with AOA, and as such a fluctuating polynomial function was selected for this, while differences in data seemed to scale with the other values directly, and thus only exponential

relations were used. Several problems emerged with this method. Camber seemed to be uncorrelated with the data, and was given an exponential value of 0. Also, because this model was created within a specific range of AOA's, the scaling factor would rapidly grow when AOA exited this range. As such, a new method was formulated.

$$F(\alpha) = [sin(\beta_1\alpha + \beta_2)sin(\beta_3\alpha + \beta_4)sin(\beta_5\alpha + \beta_6)sin(\beta_7\alpha + \beta_8)sin(\beta_9\alpha + \beta_{10})] + 1 \quad (2.41)$$

$$C_{l,Corrected} = \beta_{11} \cdot Re^{\beta_{12}} \cdot (t/c)^{\beta_{13}} \cdot F(\alpha) \cdot C_{l,XFOIL} \quad (2.42)$$

This method removed the issue of the AOA scaling factor rapidly increasing out of a certain range, by using a series of sine functions added to 1. This limited the range of possible correction values to between 0-2, while still allowing for corrections both up and down based on AOA. Camber was also removed as a factor as it produced no strong correlation with results. While this method was a marked improvement, it still had limitations. Namely, as both the α_{Stall} and α_{Cl0} value varied from airfoil to airfoil, some C_l values would increase at stall, rather than decrease as they should. To correct this, a scaled value of AOA was created between α_{Stall} and α_{Cl0}, using the XFOIL values for both.

$$A = \frac{(\alpha - \alpha_{Cl0})}{(\alpha_{Stall} - \alpha_{Cl0})} \quad (2.43)$$

$$F(A) = [sin(\beta_1 A + \beta_2)sin(\beta_3 A + \beta_4)sin(\beta_5 A + \beta_6)sin(\beta_7 A + \beta_8)sin(\beta_9 A + \beta_{10})] + 1 \quad (2.44)$$

$$C_{l,Corrected} = \beta_{11} \cdot Re^{\beta_{12}} \cdot (t/c)^{\beta_{13}} \cdot F(A) \cdot C_{l,XFOIL} \quad (2.45)$$

This final model yielded the best results of all attempts. It reduced the root mean squared error of the data from 0.1778 comparing the experimental to XFOIL data, to only 0.1189 comparing the experimental data to the corrected values, a decrease of roughly ⅓. The coefficients of this model are given below.

β_1	-0.4091
β_2	4.0339
β_3	0.4092
β_4	2.2490
β_5	-2.8047
β_6	1.7024
β_7	1.4999
β_8	-0.8916
β_9	5.5059
β_{10}	-5.9754
β_{11}	0.0655
β_{12}	0.2168
β_{13}	-0.1141

Table 2.1: β coefficient values for final MATLAB model, all unitless

Despite the usefulness of this correction, it is not without its limitations. The data it is derived from is all from Re = 17,000-60,000 and an AOA range of -5°-21°. This is outside the general design range of the project, where most Re < 20,000 and AOA can go far beyond stall. It only also limited in the type of airfoils it covers, all with (t/c) > 0.05. With these limitations in mind, the model can be treated as a general guideline for correcting C_l, and alterations to both it and the coefficients can be made to better suit the project.

Another limitation experienced here was the inability to craft a model for C_d correction. While there is a general trend of XFOIL underpredicting C_d in the pre-stall region, this is not as prominent as the trends observed in the C_l curves, and can be completely broken, as seen in the E61 at Re = 40,000 graph

in the previous section. Other sources have found that XFOIL generally predicts C_d well, especially in comparison to C_l, at low Reynolds numbers (Maughmer and Coder 2010). Because of this and C_d having comparatively lower values then C_l, effecting final performance values less, allows C_d to go uncorrected while still producing generally accurate results.

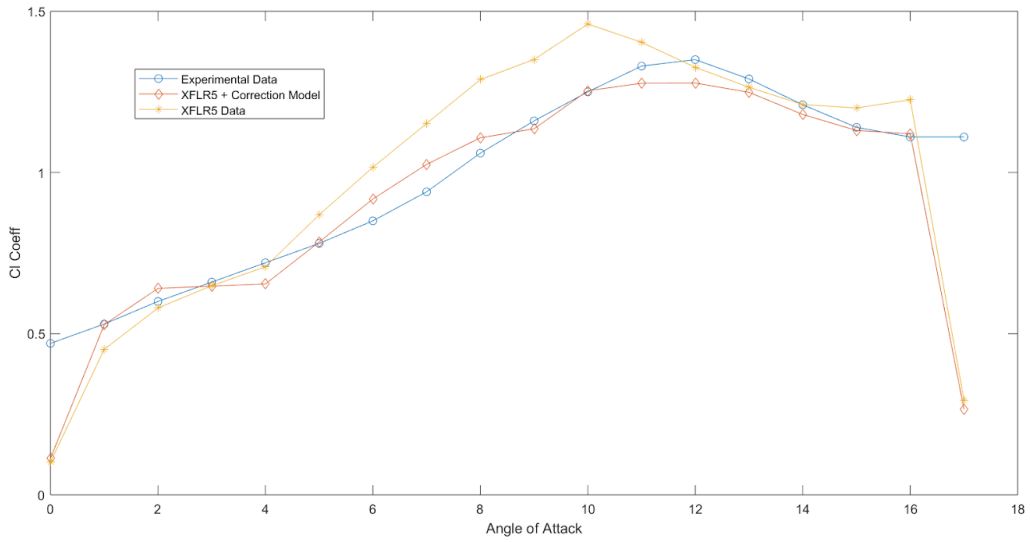

Figure 2.10: Graph Comparing XFOIL, Experimental, and Corrected results for C_l and C_d for the E61 airfoil at Re = 40,000 (Miley 1982)

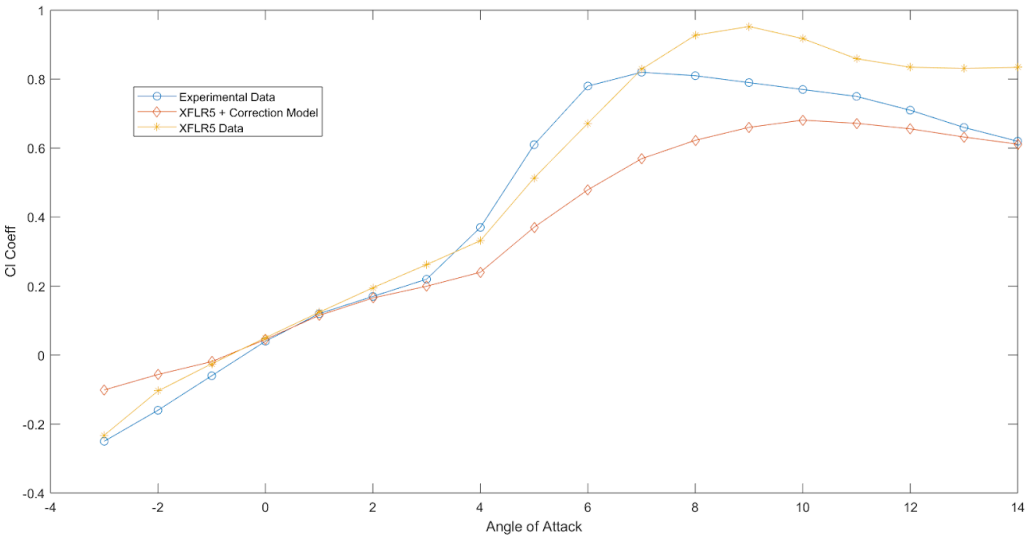

Figure 2.11. Graph Comparing XFOIL, Experimental, and Corrected results for C_l and C_d for the GOE795 airfoil at Re = 17,000 (Miley 1982)

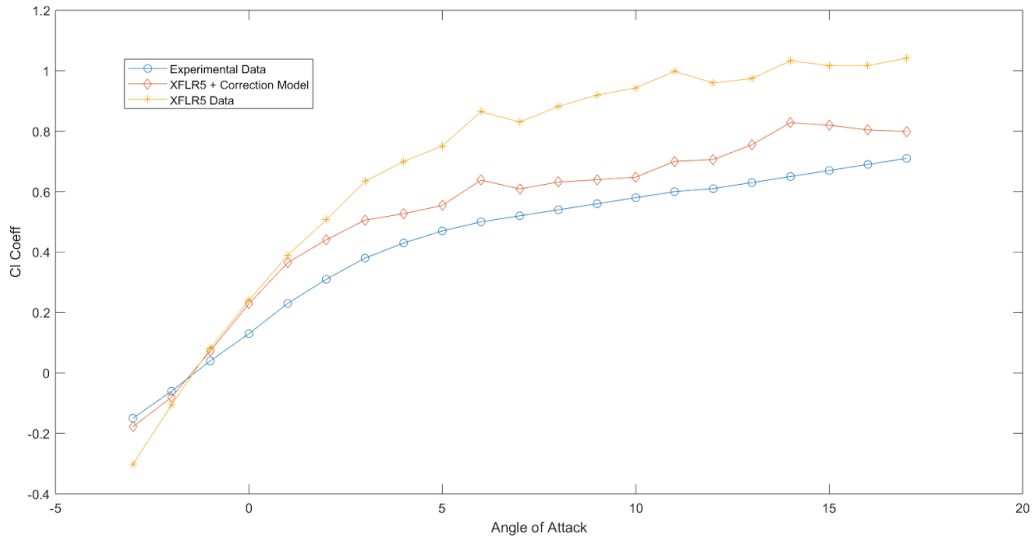

Figure 2.12: Graph Comparing XFOIL, Experimental, and Corrected results for C_l and C_d for the GOE801 airfoil at Re = 21,000 (Miley 1982)

As mentioned above, this correction was derived from an AOA range of -5°-21°, making it mostly useful for pre-stall coefficient values. This is the opposite problem of the Viterna Corrigan method, which is accurate for post stall values (Viterna and Corrigan 1982). As such, both corrections were adopted for this project, the custom correction being applied pre-stall and the Viterna-Corrigan method being used post stall. The custom correction is used to determine the new C_{ls} value, to keep continuity between the two corrections.

2.6. Simulation Data vs. Previous Study

So far, corrections to the coefficients have only been examined in the context of 2D airfoils, not at how they affect the full BEMT algorithm. To ensure the accuracy of the program, it should be tested against actual low Re experimental values to see if it can properly recreate these results. The results of 'Hover Performance of a Small-Scale Helicopter Rotor for Flying on Mars' were selected for recreation

due to the similarities to this current project (Shrestha et al. 2016). The data for this project is all gathered at Re < 5,000 over a high range of AOA, in atmospheric conditions similar to the Martian atmosphere. It also detailed much of the design of the rotor and experiment, making it an excellent candidate for recreation.

First, the conditions of the experiment needed to be recreated. These tests were carried out in 3ft diameter vacuum chamber, brought down to a pressure of 0.0167kg⁄m^3, and at an RPM range of 3000-4000. Because no specific temperature values were given, it was assumed that T = 20^{o}C, with the speed of sound being calculated from this value. Also, because it is not specified that there is a secondary pump for CO_2 gas, the composition of the atmosphere in the chamber is assumed to be the same as earths, and a corresponding value of viscosity is calculated to be 1.81x10^{-5}slug/(ft*s). Circulation effects were tested for in this experiment and found to be negligible, so this recreation will not factor them in (Shrestha et al. 2016).

The next step is to accurately recreate the rotor itself. The base dimensions are simple to recreate, with the rotor having a radius of 9”, a hub radius of 0.75”, a constant chord of 2”, and a constant pitch angle over the entire blade of 18^o-40^o. It also uses a single airfoil throughout, with t/c = 0.01 and a camber of 6.35% , all with a sharpened leading edge. These values were selected based on optimal performance at Re = 50,000. It should be noted that chordwise position of maximum thickness and chord are not specified and are thus both assumed to be 0.5c. Using these specifications, this airfoil design was recreated in XFLR5 and polars were found for it at Re = 1,000, 3,000, and 5,000. In all instances Mach number was assumed to be 0 and the N_{CRIT} was kept at its default value of 9.

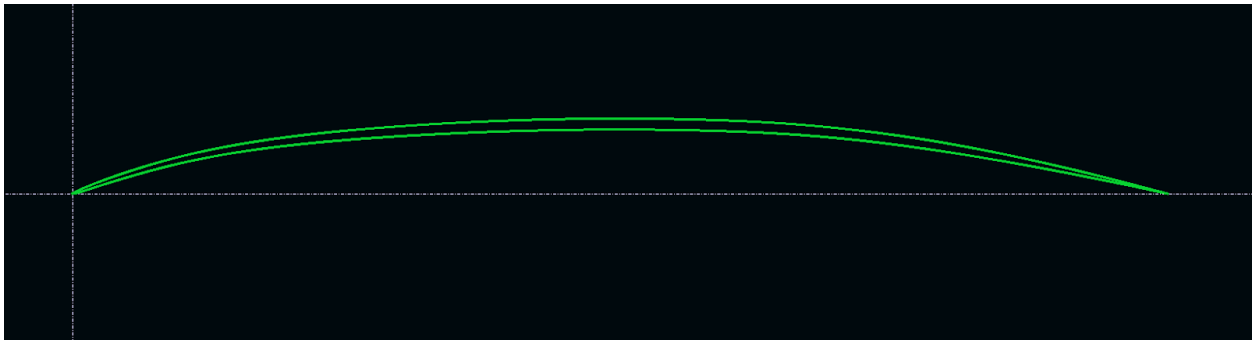

Figure 2.13: Cross Section of the recreated airfoil

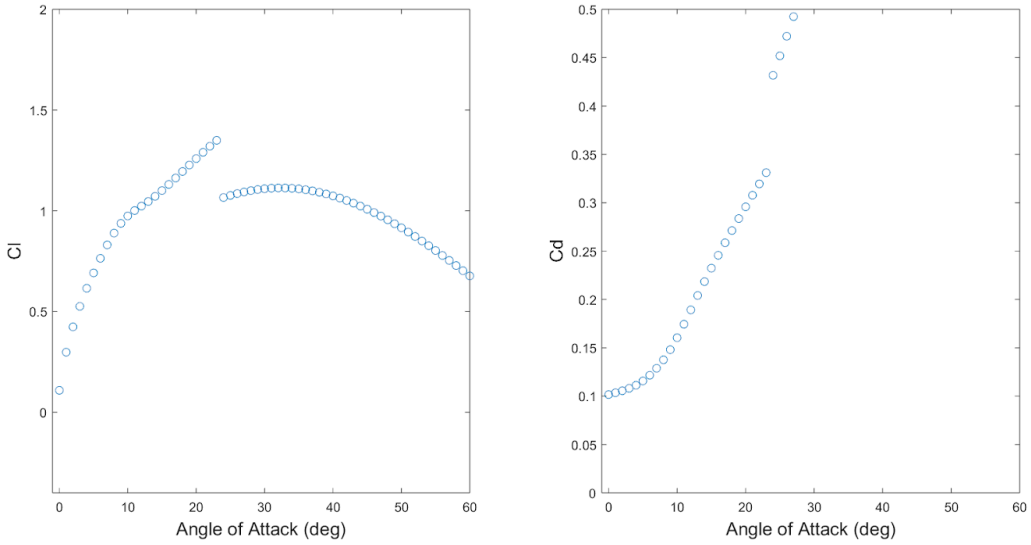

Figure 2.14: XFOIL Polars for the recreated airfoil at Re = 1000

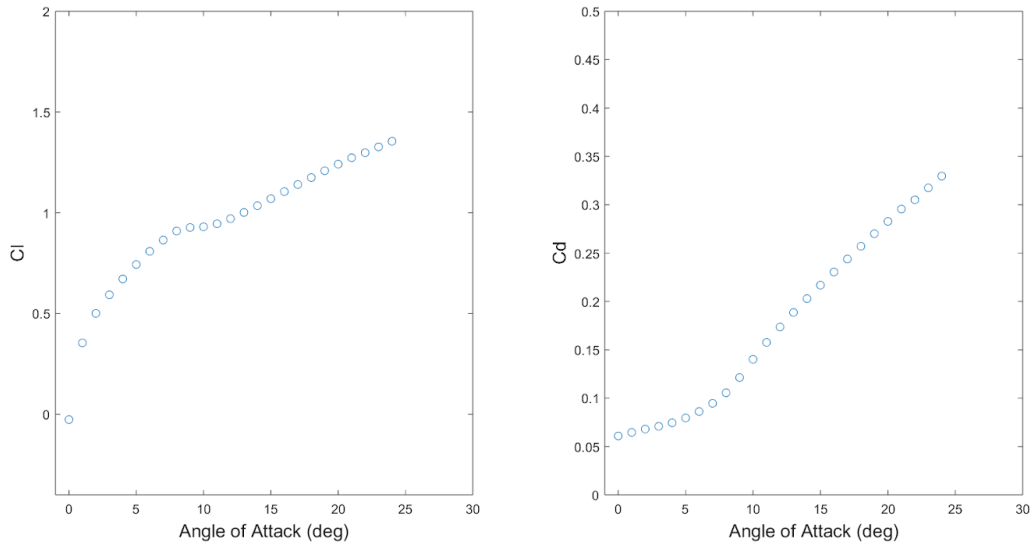

Figure 2.15: XFOIL Polars for the recreated airfoil at Re = 3000

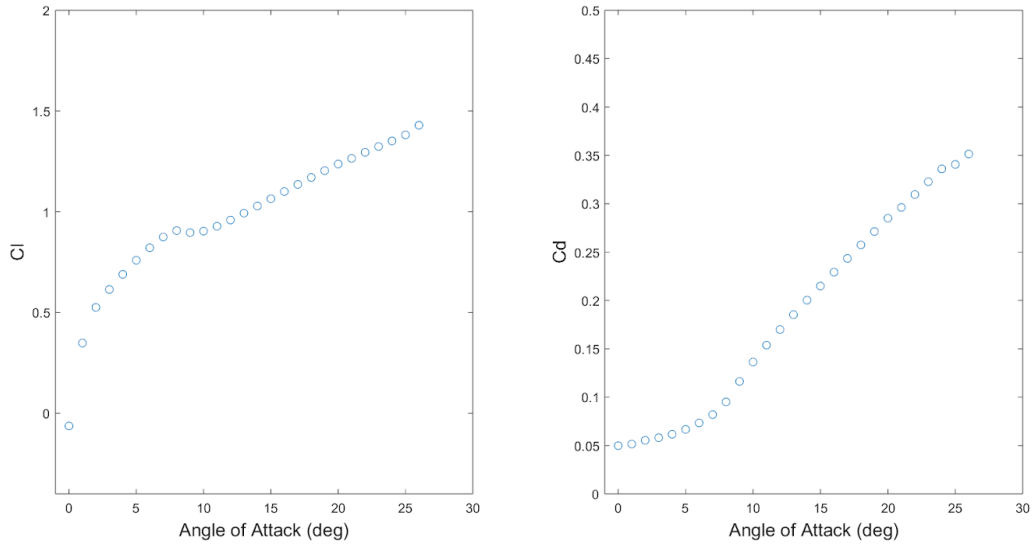

Figure 2.16: XFOIL Polars for the recreated airfoil at Re = 5000

With the polars generated it is important to assess them for general accuracy and trends, particularly because the correction models require information from both C_{l0} and stall. Chief amongst these aspects is the general shape of the curves. C_l polars at low Reynolds numbers have particular shapes they tend to adhere to based on their t/c and camber values. Of particular interest in these

instances are the drop and recovery patterns, which occur at moderate t/c and low camber for recovery and high t/c and cambers for drop (Lee et al. 2006). Although this foil doesn't explicitly replicate these attributes, they all show a recovery directly after stall, and the Re = 1000 polar shows a significant drop later on. It can be assumed that the other polars would likely have seen a similar drop if they had been able to converge at higher AOA. It is likely the low thickness of the airfoil, moderate camber, and hooked trailing edge that lead to these patterns occurring (Lee et al. 2006). It should also be noted here that the recovery style curve is exactly what the Viterna-Corrigan model excels at recreating, meaning these polars are encouraging to observe (Viterna and Corrigan 1982; MacNeill and Verstraete 2017).

To assure the accuracy of the simulation, the point of stall must be found. Looking at the curve for Re = 1000, it may be assumed that stall occurs at the point where there is a drop. This, however, is out of step with both the other two graphs, which show stalls at far lower AOA and far lower C_l values, and also with previous knowledge of how polars act in these low Re regimes. Previous studies show thin airfoils, such as the NACA0002 experiencing stall below 10^o and with $C_{lMAX} \approx 0.5$ at similar Re values, making it unlikely that the foil stalls out past 20^o at $C_{lMAX} = 1.35$ (Kunz 2003). A small decrease in the slope of the curve is observed around the same point where the other polars experience stall, making this the most likely place where stall occurs. With this assumption we see a trend apparent in other foils at these regimes, where stall angle is pushed to lower angles at higher Re values, while C_{lMAX} also decreases (Kunz 2003)

Reynolds Number	α_{Cl0} (deg)	α_{Stall} (deg)	$C_{l,STALL}$
1,000	-0.5	11.5	1.0122
3,000	0.1	9.4	0.9283
5,000	0.2	8.0	0.9066

Table 2.2: Stall and zero lift data for the recreated foil

With the data matching expected patterns, and the conditions set, the BEMT algorithm is ready to use to recreate this experiment. This follows the same procedure outlined in sections 2.2 and 2.3, with the corrections taking place whenever the C_l and C_d values are derived. It is at this point where the deficiencies of the correction can be examined. Of particular interest are coefficients β_{12} and β_{13}, which deal with how much the Reynolds number and t/c affect the overall correction. The model was derived using data from Re = 17,000-60,000 and t/c = 0.0567-0.16, considerably higher than the values of Re < 5,000 and t/c = 0.01 used in this experiment. As such it is reasonable to assume that the correction may underpredict the full extent that these values must be corrected by. Of particular note is the fact that β_{13} = -0.1141, which means that as the airfoil grows thinner, the more accurate XFOIL becomes. However, as previous studies noted, XFOIL produces its best predictions when there is a steep pressure recovery gradient on the upper surface of the airfoil (Maughmer and Coder 2010). This type of pressure gradient occurs on thicker airfoils rather than thinner, leading to the conclusion that accuracy should drop with thinner airfoils (Kunz 2003). There is also the issue that all Reynolds Numbers used in making the model are all an order of magnitude higher than the ones used in the experimental recreation, meaning its exponential factor could be inaccurate.

With these considerations in mind, multiple simulations were run, changing the values in order to find the best possible combination. β_{13} was set mainly as a positive value, to better reflect the known pattern, and β_{12} was generally tried at values < 0.2, to account for the greater probable error in this regime. It was found that certain combinations would raise accuracy at lower pitch angles, while raising it at higher values. The opposite relation also occurred. This leads to the conclusion that there is a likely AOA dependence for these coefficients, however, upon analysis no specific relation could be found. A combination of β_{12} = 0.02, and β_{13} = 0.07 were found to work best as a compromise in accuracy between these two extremes.

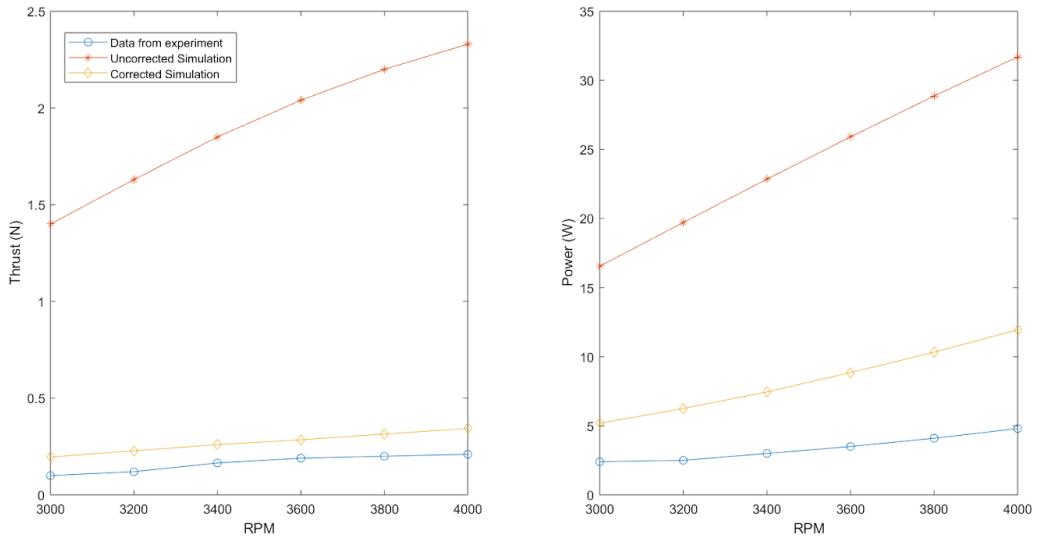

Figure 2.17: Thrust and Power Graphs showing the Corrected and Uncorrected Sim data compared to experimental values at pitch angle = 18°

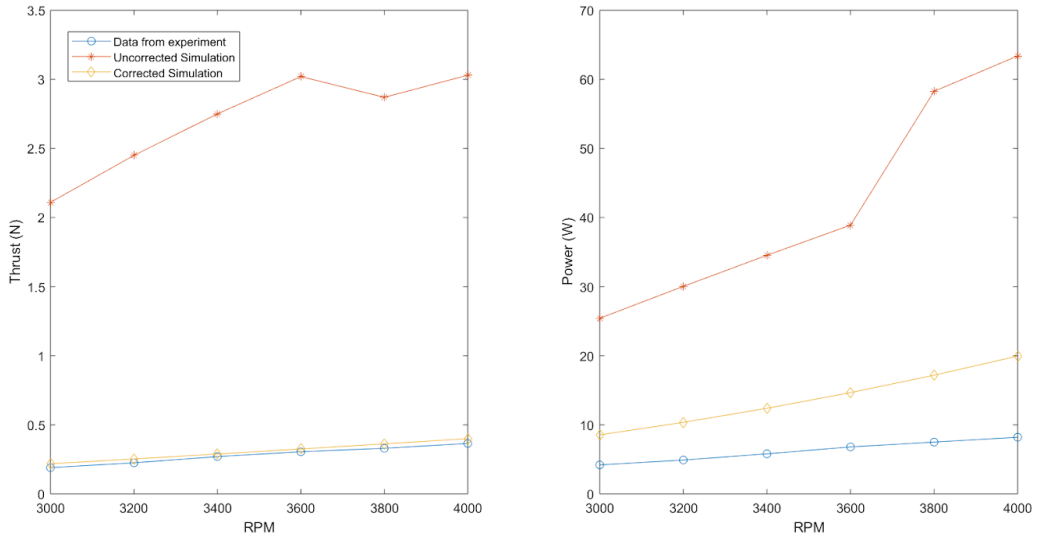

Figure 2.18: Thrust and Power Graphs showing the Corrected and Uncorrected Sim data compared to experimental values at pitch angle = 28°

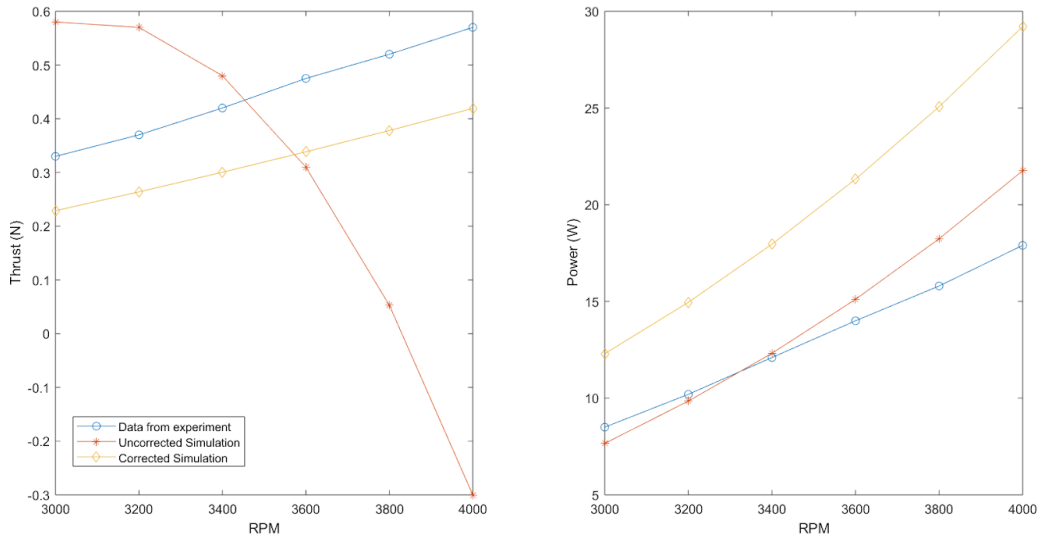

Figure 2.19: Thrust and Power Graphs showing the Corrected and Uncorrected Sim data compared to experimental values at pitch angle = 38°

The first observation we can make is that, on almost all graphs, the simulation with corrections comes far closer to the experimental values then the uncorrected values, especially with regards to thrust. This makes sense, as thrust is far more lift dependent then power, meaning a correction to lift will lead to more noticeable results. The power estimates also improve, being better than the uncorrected value at all pitch angles except for 38°. However, looking at the thrust results at this angle, it is likely that the uncorrected power values are coincidental. The power estimate could likely be improved on with better corrections to drag. Another notable result seen here is that the uncorrected values all overpredict their performance metric, consistent with the previous observations of lift being over predicted by XFOIL.

This is a single experiment being recreated, and thus is not a complete confirmation of the BEMT simulation. It fails to account for possible surface roughness of the blades, or possible bending during rotation, both of which would have a significant effect on the final results. However, it does offer physical data validation and allows the corrections to be calibrated to proper values. With this method, the

project can advance with more confidence in its preliminary numbers, and better knowledge of its limitations.

CHAPTER 3

THE DESIGN PROCESS

With a sufficient simulation in place, the design process can begin. Creating a propeller for the Mars Atmosphere poses several challenges and issues which need to be accounted for. The airfoil choice, number of blades, radius, blade twist, and chord distribution all have different effects in this Reynolds number regime than they may have in others. Separate considerations must be made for all of these variables, based on previous experimental data and design attempts. These decisions will be further refined in the optimization section.

While there are several different functionalities the propeller can be designed for (take off, lateral motion, direction changes, etc.) this project will primarily focus on designing and validating for hover conditions. The reasoning behind this choice is due to the lack of known conditions for this design. There is no set payload, no desired mission length, no achievable altitude. As such, focusing purely on the hover performance is the simplest place to begin, and the data gathered here can be used to select appropriate payloads and missions for this propeller.

3.1. Design Challenges and General Principles

When starting the design process it is important to identify the main goals of the design, the parameters for its operation, and the challenges that will come from these. The broadest goal of this project is to create a propeller that will allow for the take-off and hovering of a UAV in the Mars atmosphere. This means that a primary focus should be placed on generating sufficient lift for these tasks. Beyond this, a secondary goal should be to create an efficient design, allowing the drone to operate for longer periods of time, over greater distances. As a note, efficiency for this paper will be defined as the thrust to power ratio of the propeller, not the typical definition for propeller efficiency. This is because the propeller is being designed for hover conditions, and thus no forward airspeed. Using the traditional definition of propeller efficiency would involve the advanced coefficient, J, which would

always be at 0 due this lack of forward airspeed. As such, the thrust to power ratio is a more effective metric for this project. Other concerns typically taken with propeller design are of less importance here. Sound caused by the propeller is a nonfactor, as it is set to operate on Mars and not near any organism that could be harmed by excessive noise. Icing is also of lesser importance, as Mars's atmosphere contains far less water vapor than earths, anywhere from 10-100 times less than the driest parts of earth (Davila and Schulze-Makuch 2016).

With these two design goals in mind, the obstacles to achieving them can now be examined. Looking at Equation 2.13, it can be seen that thrust is highly dependent on the individual lift components, which are themselves dependent on velocity, chord length, radius, C_l, and density. Of particular concern for this project is the density. The ambient density on Mars surface found in the model being used here is 0.01397 kg/m^3 while the standard density for Earth at sea level is typically given as 1.225 kg/m^3, roughly a 100-fold difference (Colozza et al. 2005; Anderson 2012). This is the equivalent of the Earth's atmosphere at an altitude of roughly 31.5 km (Anderson 2012). This can be improved slightly with compressibility effects, but other improvements to the propeller design are necessary to compensate for this large drop in density. These improvements can be derived from the propeller RPM, chord length, radius, blade twist, and aerodynamic coefficients, although all of them come with a variety of setbacks.

One of the most prominent and impactful of these trade offs is the interplay between RPM and blade radius. Higher RPM and longer blade radius both contribute to higher thrust, but can also increase the drag and power requirements for the propeller. Perhaps more importantly, there is also the hard limit of the tip Mach number of the propeller. At excessively high Mach numbers, shockwaves can form along the blade, increasing drag, inducing flutter, and possibly damaging the propeller (Colozza 1998). Many propeller designers will set a hard Mach number limit at M = 0.75 in order to avoid the formation of these shockwaves. This lines up well with the compression correction used in the BEMT algorithm

being limited to M < 0.7-0.75. Mach number at the tip is a product of both radius and RPM, and thus can be calculated from the two values, although this would not account for other factors such as induced velocity. Also of note when calculating Mach number are the effects that temperature and atmospheric composition have on the speed of sound. Because the Martian atmosphere is composed of roughly 95% CO_2 (Colozza et al. 2005), the atmosphere can be assumed to be an ideal gas of purely CO_2 when calculating the speed of sound. With the lower adiabatic constant and higher molar mass of CO_2 in comparison to air on earth,(Miley 1982)) and also considering the lower temperatures expected on average on Mars (Colozza et al. 2005), it can be assumed that the speed of sound on Mars would be far lower than that observed on Earth (White 2016). When calculated at 'sealevel', the speed of sound on Mars is found to be 240 m/s or 787.5 ft/s, significantly smaller than the 340 m/s or 1115.5 ft/s it would be on Earth. Using this value, and a limit of M = 0.7, a curve can be plotted detailing the viable range of radii and RPM that can be used.

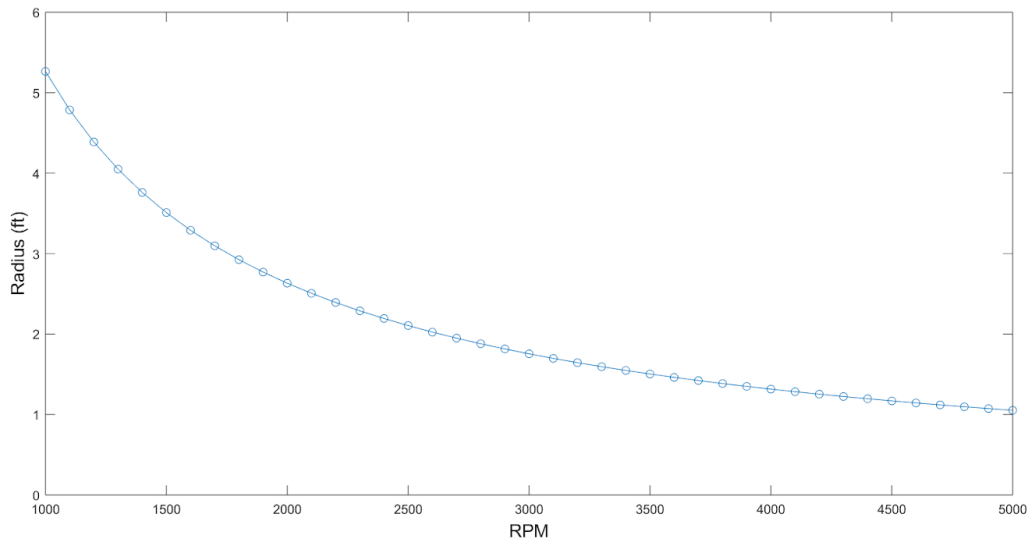

Figure 3.1: Radius vs. RPM curve, plotted at M = 0.7. Below the curve is the viable design space

Looking at the chart, there is clearly a choice to be made between high radius and low RPM or the converse. These naturally come with their own benefits and drawbacks, both of which need to be

assessed during the design phase. One of the most prominent effects from this decision is the change in Reynolds number that can occur. If there is a lower radius, then RPM can naturally grow higher. As a result, this raises the Reynolds number at every blade station along the blade. The effects of higher Reynolds number are discussed above, but can lead to a variety of positive effects such as higher C_{lMAX}, as well as higher C_l/C_d. This would accomplish both goals of the propeller design, generating greater thrust, while becoming more efficient.

One study examined this analytically and found that while the higher RPMs can lead to better performance, in general the reduction of diameter leads to lower thrust and efficiency (Colozza 1998). This makes some sense, although the study does not specify if RPM or M are kept constant in its analysis, and the general causes of these reductions in performance are not examined. Assuming that RPM is kept constant, it is natural that thrust would decrease with a reduction in radius, although the effects of this could be compounded if the chord is similarly scaled with the radius. Efficiency likewise may suffer from this reduction in thrust, while torque could remain high due to the lower Reynolds number values increasing drag, once again assuming a constant RPM. If the Mach number is kept constant instead, these results become more dubious, especially without deeper analysis.

Perhaps of more value are experimental results, of which there are several, mostly at smaller scale. One such study, conducted by the University of Illinois examined small scale propeller performance at Re = 50,000 - 100,000 (Brandt and Selig 2011). While these are all done at higher Reynold numbers then this design process, and generally on smaller propellers, the results are illustrative of what this project may experience. It is found, repeatedly, that the thrust coefficient and efficiency of the propeller rises at higher RPMs, likely due to the Reynolds number effects examined above. Also, this trend seems to continue with lower radius, but similar RPMs, contradicting the analytical results of the previous study. Of particular interest for this study would be the ‘slow flyer’ propeller models, as they are more likely to imitate the low tip speed ratio conditions that a Mars UAV

propeller would experience. The APC Slow Flyer propeller shows these relationships well, with better performance metrics at higher RPMs and lower radii, however the GWS shows more mixed results. Both thrust coefficient and efficiency change with RPM and radius, but neither show a clear trend. Other, non-'slow-flyer'models such as the Graupner CAM Slim, show trends more similar to the APC propeller then the GWS, making it likely that it is simply an outlier. This is further reinforced by a follow up study performed with more propellers in the same facilities, which came to the same conclusion of higher RPM and lower radii leading to better performance metrics (Deters, Krishnan, and Selig 2014).

Radius and RPM are not the only parameters that can be manipulated to account for better performance; however, they are easier to examine individually. The remaining factors such as aerodynamic coefficients, chord, twist, and pitch of the propeller are all highly dependent on other factors, and each other. General trends that may affect one propeller may not be applicable to another, because of these changes. To fully examine what effects they may have, some decisions should be made beforehand. Chief amongst these is the selected airfoil used for the propeller. All of the listed factors are dependent on this choice, and how best to optimize for them is directly related to this decision.

3.2. Airfoils at Low Reynolds Numbers

In order to continue the design process, a decision needs to be made about the choice of airfoil for the propeller. Discussion of flow at extremely low Reynolds Numbers has been done in section 2.3, but how these conditions specifically affect airfoils bears further analysis. There is both the theoretical background to examine, as well as previous designs to look at. This study will utilize both to create an airfoil which best suits the needs of the project.

Much of the design of airfoils at low Reynolds Numbers has occurred at far higher Re values than the one examined by this project, usually for Re > 50,000. With this in mind, many of the principles used by these designs may be applicable to airfoils at lower Re, so they can't be entirely discarded. One design of particular interest is the S1223 airfoil. This design was created as part of a broader design

philosophy, focused on making high-lift airfoils for relatively low Reynolds numbers. In this instance, the assumed application would be for smaller UAVs and the assumed Reynolds number was 200,000 (Selig and Guglielmo 1997). During this design process, they identified one key component that many higher-lift airfoils exhibit at this regime which is the aft-loading of higher cambers. By shaping the airfoil like this, designers can take advantage of the concave pressure recovery that many airfoils already possess and enhance the effects of it with an added pitching moment (Selig and Guglielmo 1997; Eppler 2012). This design principle has been utilized by several other airfoils, such as the FX 74-CL5-140, although the effects that it may have on airfoils at Re < 10,000 may be limited. As was noted in section 2.3, pressure recovery at these Reynolds numbers is almost non-existent, making the possible effects of aft-loading unclear (Kunz 2003).

For a better understanding, airfoils at lower Reynolds numbers should be examined, specifically with regard to their physical properties and what effects they have on performance. This has been done several times in the past, typically with a focus on insect wings as these naturally fly at Re < 20,000. One such study examines both flat plates and various airfoils at Re = 11,000-15,000, with a particular focus on the effects of certain physical properties (Okamoto, Yasuda, and Azuma 1996). For instance, it was observed that as the thickness of a flat plate decreased, performance improved with both higher C_{lMAX} values and lower C_d values, in addition to lower lift curve slopes. This increases thrust, efficiency, and delays stall, all great improvements. With regards to camber, the study found that increasing it, specifically in an upwards convex style, increases C_{lMAX}, C_{dMIN}, and lift slope, while also pushing back the stall angle. A higher C_{lMAX} and delayed stall angle are also the result of sharpening the leading edge of the airfoil, although other studies show minimal difference and even lift penalties for a sharpened leading edge (Kunz 2003). A more major point of disagreement between this study and others is the effect of aft-loading the camber. In this study it is observed that when moving the maximum camber back along the chord line, both the lift and drag coefficients decrease slightly, although stall is further pushed back.

This runs contradictory to previously established theory and later studies. One explanation for the lack of increase in C_l and C_{lMAX} is that the flow conditions at these Reynolds numbers, with their flat pressure recoveries, may not be able to take advantage of the aft-loading the same way that higher Re airfoils are able to. Apart from these traits, the study also examines the effects of surface roughness and corrugation of airfoils, similar to the shape of insect wings, and while they produce some positive results it is unclear how they might work with rotational motion (Okamoto, Yasuda, and Azuma 1996). As such, these observations are likely not of use to this project.

There have been other studies of a similar nature. One such study examined various airfoils at Re = 4,000, again focusing on the effects of different physical characteristics (Shyy et al. 2007; Sunada et al. 2002). A notable observation made by this study is about the effects of camber on performance. It mostly agrees with the previous study, observing that raised cambers lead to increased lift slopes, higher C_d values, and delayed stall. However, it observed that when camber was pushed towards the trailing edge of the airfoil, both C_l and C_d values increased, a direct contradiction to the results of the previous study. The causes of this discrepancy are unknown, but could be due to different Reynolds numbers in the experiments or simply differences in the experimental set ups.

Another study used theoretical calculations based on an Incompressible Navier-Stokes Solver with Viscous-Inviscid Interaction Methods and some flow field assumptions, to compare the traits and effects of various NACA foils at Re < 10,000 (Kunz 2003). Calculating aerodynamic coefficients for NACA 0002-0008 airfoils, to see the effects of thickness, the study found an increase in drag coefficients and a reduction in C_l for the thicker airfoils. This reduction in performance is only exacerbated the lower the Re of the flow is. While mostly in line with the results of the first experimental study, these results do contradict with regard to the effects of thickness on lift curve slope. The theoretical study observes, due to greater viscous effects at low Re values, that as thickness increases it effectively reduces camber and the perceived AoA of the airfoil, thus lowering the lift curve slope for thicker models. The first study finds

the opposite of this phenomenon, with thinner wings having the lower slopes, although this relationship is minor and at Re > 10,000 (Okamoto, Yasuda, and Azuma 1996).

In addition to the thickness, this study examines the effect of camber, specifically using the NACA 0002 and NACA 4402. The general effects of added camber are that the zero-lift AOA is pushed back, while lift and drag receive large and modest increases, respectively (Kunz 2003). Because lift increases at a higher factor with changing camber, the L/D ratio generally also sees improvement. This effect might reverse, however, at higher cambers, as observed later in the paper. It is also observed that the addition of camber delays trailing edge separation at lower Reynolds numbers, a favorable result for performance. The effects of maximum camber position are also observed using foils with 2-4% camber at 30, 50, and 70% chord position for the maximum value. Here it is found that aft-loading the camber results in higher C_{lMAX}, stall angles, and L/D, in agreement with the second study mentioned (Kunz 2003; Sunada et al. 2002; Shyy et al. 2007). In general this study recommends having a moderate camber and aft-loading for better performance, with higher cambers at lower Reynolds numbers to account for the higher viscous effects in these regimes (Kunz 2003).

Study	Blade Thickness	Size of Camber	Camber Position
Okamoto, Yasuda, and Azuma 1996	Thinner airfoils lead to lower drag and higher maximum lift. Lift curves have lower slopes	Increased camber leads to higher lift slopes, drag, and stall angles.	Aft-loading the camber causes lower lift and drag slightly, while delaying stall.
Sunada et al. 2002 and Shyy et al. 2007	n/a	Increased camber leads to higher lift slopes, drag, and stall angles.	Aft-loading the camber causes higher lift and drag values.
Kunz 2003	Thinner airfoils lead to lower drag and higher lift. This effect grows larger at smaller Re. Lift curves have higher slopes.	Increased camber leads to higher lift and drag, and delays trailing edge separation.	Aft-loading the camber causes higher maximum lift, delays stall, and improves efficiency.

Table 3.1: Airfoil attributes and effects, based on studies (Okamoto, Yasuda, and Azuma 1996; Sunada et al. 2002; Kunz 2003; Shyy et al. 2007)

3.3. Design of Final Selected Airfoil

Considering the results of the previous section, certain airfoil traits can be selected due to their favorable thrust and efficiency. One very prominent trait is keeping the foil thin. This raises lift and lowers drag, leading to both better thrust and efficiency, and can also possibly lead to stall delay and a greater range of usable AOA (Okamoto, Yasuda, and Azuma 1996; Kunz 2003). Ideally the design would lower thickness as much as possible, but, due to concerns about structural integrity and possible damage due to fluttering effects, there is likely a lower limit on blade thickness. Without greater testing and knowledge of materials, this limit can not be properly assessed, and thus an arbitrary lower limit of t = 0.03c will be used. This can naturally be revised and changed, as more is learned about the building materials and mission of the UAV.

The other most favorable design choice is to have a moderate, aft-loaded, camber. Although some studies have contradicted the effectiveness of this design, the majority show major gains in C_l,

enough to outpace gains in C_d and thus raise the overall L/D (Selig and Guglielmo 1997; Okamoto, Yasuda, and Azuma 1996; Kunz 2003). There may, however, be a limit to this effect based on camber size, with certain studies observing reduced performance at camber > 5% (Kunz 2003). As such, the chosen design will likely try to be below this value.

Several airfoil designs were examined in XFLR5, some pre-established, some original, and some based on previous designs. Attention was paid particularly to their lift and L/D characteristics, with other metrics such as drag and stall being secondary considerations. While many airfoils were tested, a sampling of the more successful and interesting designs are shown below.

One such example is the E61 airfoil, designed by Richard Eppler. This airfoil meets all of the general requirements the desired foil would require, being relatively thin at 0.0567c thickness, and a 6.69% camber loaded more towards the trailing edge. It had also been previously tested in low Reynolds Number environments, making it a good candidate for this design (Miley 1982). When examined in XFLR5, this airfoil was found to have both a high stall angle, and a relatively good efficiency, although this is obviously highly dependent on Reynolds number. One area in which the design was found lacking was its relatively low C_{lMAX}, which was notably smaller than the others. This naturally could lead to issues with achieving the necessary thrust for the design.

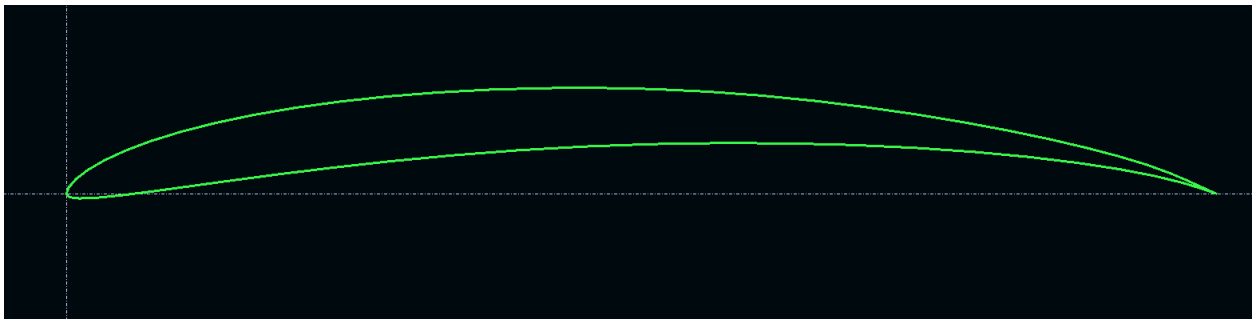

Figure 3.2: E61 Airfoil Cross Section

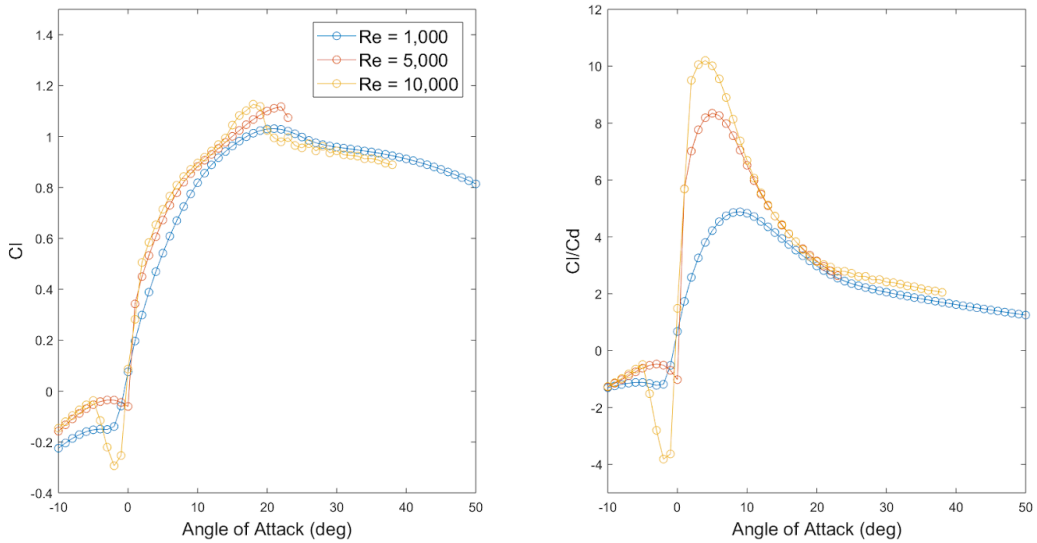

Figure 3.3: E61 Airfoil Polars at Re = 1,000-10,0000

Custom airfoils, based on the criteria outlined above, were also examined. These all generally followed the mold of thin foils with medium aft-loaded cambers, although the degree to which they did so varied from foil to foil. Other traits, such as sharpened or flattened leading edges, were also tested. Despite the differences in design, the XFLR5 results either held steady through all designs or failed to converge. An example is seen below in Figures 19-20. In general these foils were found to have the greatest efficiency of all tested designs, in line with previous experimental results, although both the stall angle and C_{lMAX} values were lower than the other designs. It should be noted that these foils exhibited excellent post-stall lift recovery, achieving very high C_l values, although the accuracy of these results are uncertain.

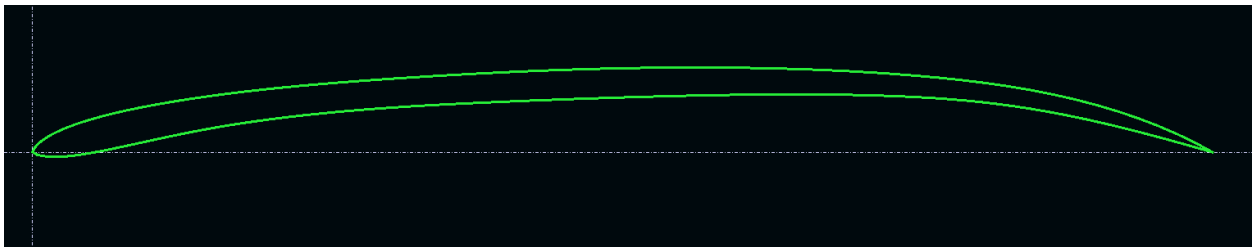

Figure 3.4: Custom Airfoil Cross Section

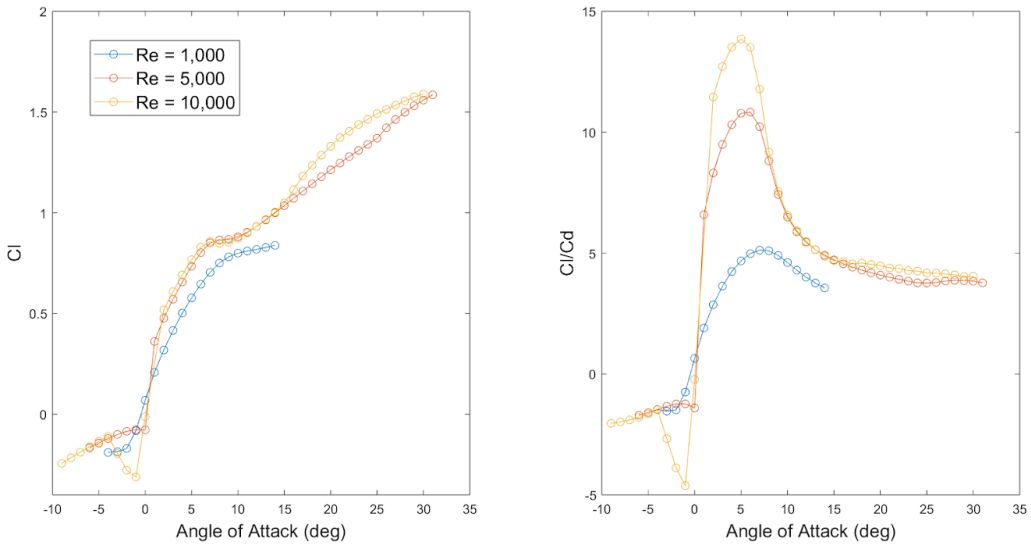

Figure 3.5: Custom Airfoil Polars at Re = 1,000-10,0000

The final set of airfoils examined were existing designs for low Reynolds Number environments that were then modified to better reflect the above criteria. These had various results, dependent on what their original design was, with the best result coming from the S1223 airfoil, with a modification bringing the maximum thickness down to t = 0.05c and a camber of 7.45%. Its efficiency is marginally higher than the results from the E61 airfoil, although it did notably lag behind the custom foils. Where it excelled, however, is with its lift coefficients. It possesses a very high stall angle, with a high C_{lMAX} at this point. There does appear to be a sharp decline after stall, although as pointed out before, XFLR5 can be inaccurate at modeling post stall coefficients in this regime.

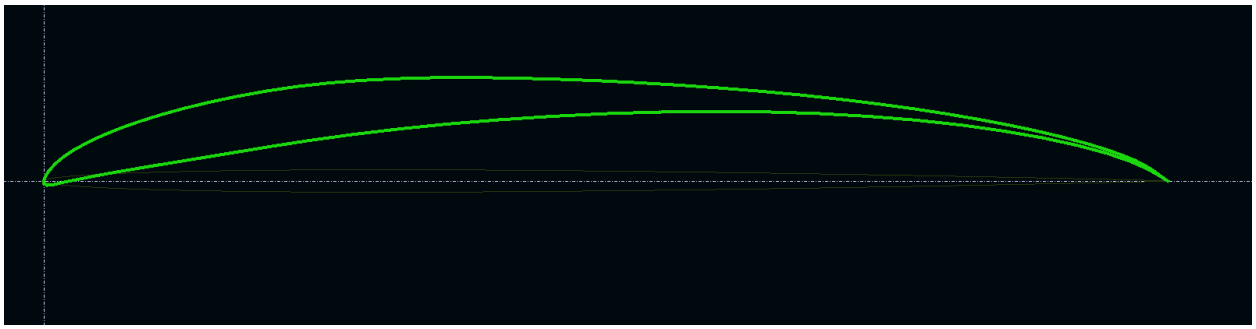

Figure 3.6: Modified S1223 Airfoil Cross Section

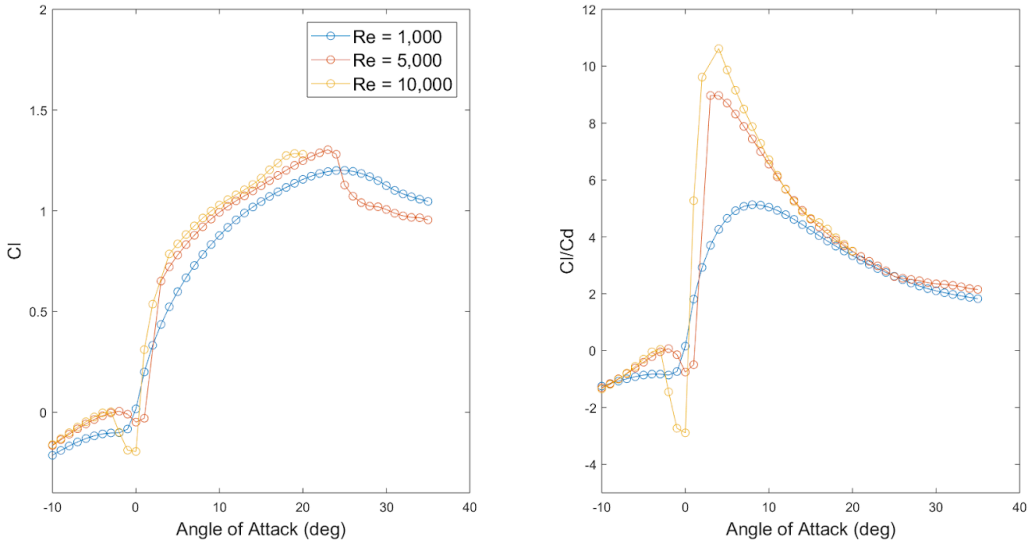

Figure 3.7: Modified S1223 Airfoil Polars at Re = 1,000-10,0000

From all of these possible choices, the modified S1223 was selected. In addition to the positive attributes listed above, it is also among the thicker airfoils examined during this process, giving it a higher durability against bending and flutter. More confidence can also be placed in the XFLR5 values, as the design is based on an airfoil specifically created for high lift in low Reynolds Number environments.

Reynolds Number	α_{Cl0} (deg)	α_{Stall} (deg)	$C_{l,STALL}$
1,000	-0.1713	24.52	1.203
4,000	1.0317	25.93	1.313
7,000	0.8174	21.94	1.278
10,000	0.3844	18.33	1.287
13,000	0.3402	18.30	1.317
16,000	0.3012	19.73	1.364

Table 3.2: Stall and zero lift data for the modified S1223 foil

Here we can see the stall and lift parameters of the modified S1223 airfoil, over the general range of Reynolds Numbers being designed for, derived from the XFOIL code. These values were all

determined using linear interpolation from their XFOIL data. All zero lift angles are between -0.2°-1.1°, a small range, with possible error due to the XFOIL's calculations of C_l for low Re. The data also, generally, follows the trend of decreasing stall angles at higher Reynolds Numbers, with small exceptions at Re = 4,000 and 16,000. Observed stall angles are higher than other foils examined. However, this design is specifically geared towards pushing back stall, and experimental data shows the foil having high stall angles even at considerably higher Re values (Selig and Guglielmo 1997). The C_{lMAX} values show a general trend of increasing with higher Re values, with a notable exception at Re = 4,000. Previous studies had shown an opposite effect, with C_{lMAX} values decreasing at higher Re, although these only looked at Re < 6,000 (Kunz 2003). This means that the drop in C_{lMAX} may only be a temporary phenomenon, reversing past a certain Re value. As all these values fall within a small range of 0.161, and lacking experimental data for a basis, these results are reasonable and can be used as the basis for the BEMT simulation.

3.4. Pitch Angle, Chord Distribution, and Other Considerations

With the desired airfoil in place, final selections can be made about the propeller shape, namely the pitch angle, twist, and chord distribution. Pitch angle is heavily dependent on both the airfoil and intended mission of the propeller. While it could be simply selected for whatever angle it experiences maximum lift at, this fails to account for induced velocity and the possibility of greater post-stall recovery. As such final decisions about the pitch angle will be made during the optimization phase, which can account for these factors.

Determining the twist does not require such considerations. This is typically based on the assumed mission of the propeller, being optimized for whatever average airspeed the propeller will encounter. As the intended condition for this propeller is hovering, meaning zero forward airspeed, then there should be no twist for the blade. This also fails to consider the induced airspeed of the propeller, but it is small enough compared to the rotational speed of the propeller that this possible loss in efficiency is negligible.

This leaves the chord distribution. There are countless ways to distribute the chord radially along the propeller, depending on different conditions and objectives. In the experiment examined in section 2.6, a constant chord propeller was used (Shrestha et al. 2016). This is the simplest option, but comes with several drawbacks. Most prominently, the increased chord at the tip of the blades creates a larger drag force at the tips and thus a much larger torque, making for a less efficient propeller. In addition to this, the higher force nearer the tip can lead to blade deformation and possible structural damage. The natural fix for this is to decrease the chord size at the tip, although there are countless methods of doing this.

For this propeller, the Betz Optimal Rotor distribution was utilized. This distribution is derived from blade element and momentum theory, and was derived specifically to achieve the Betz Power Limit for wind turbines (Manwell, McGowan, and Rogers 2009). Several assumptions go into this, such as neglecting drag force and performance losses from finite blades. Furthermore, it requires some knowledge of the propeller design such as radius, number of blades, tip speed ratio, and data for the chosen airfoil. This propeller shape also has a formula for blade twist, on which the chord distribution is dependent. However, as stated above, this design will not utilize blade twist, and thus this part of the formula will simply be used to determine the chord, but not be used for the actual design.

$$\lambda_r = \lambda\left(\frac{r}{R}\right) \quad (3.1)$$

$$\varphi = \frac{2}{3} tan^{-1}\left(\frac{1}{\lambda_r}\right) \quad (3.2)$$

$$c = \frac{8\pi r}{BC_l}(1 - cos(\varphi)) \quad (3.3)$$

As this is a propeller intended to generate lift, not transfer wind into power, there are naturally some issues with using this formula. The most noticeable issue comes with the effect of tip speed ratio. This is a ratio defined by the rotational speed of the propeller divided by the forward airspeed. For a

wind turbine, this value is typically very low, whereas a hovering UAV propeller will have an extremely high value. Due to the inverse relationship between tip speed ratio and the chord length, the chord will grow shorter as the ratio increases. This is antithetical to the goal of generating sufficient thrust in a low density atmosphere. As such, adjustments must be made. Either an artificially low tip speed ratio should be used, or a linear multiplication factor can be applied after the fact. For this design, both methods were used.

To establish the base chord distribution, certain numbers needed to be known, specifically the radius, blade number, tip speed ratio, and a specific C_l value. As the chord scales with radius, any value can be used here. The number of blades was kept at 2, mimicking designs seen in both the recreated experiment as well as the NASA MARS2020 project (Shrestha et al. 2016; Northon 2018). The C_l value used in the calculation was taken at the point of maximum C_l/C_d to best account for the assumption of no drag (Manwell, McGowan, and Rogers 2009). Finally, for tip speed ratio, a value of 9 was selected, higher than what is necessary for the chord, but considerably lower than what the actual value would be.

This yields a good base to begin, but alterations still need to be made. Most significantly, alterations need to be made near the hub of the propeller. As the chord is largest here, it needs to be reduced significantly in order for it to attach to a reasonably sized hub. On this assumption, the chord where the blade meets the hub should be no greater than the diameter of the hub itself. This adjustment is done using Excel graphing and a best fit curve, to give a visualization of what the blade will look like. Once the blade is shaped reasonably, the equation from the best fit curve can be brought into the BEMT program to give the chord of the blade. The equation used in this instance is:

$$\frac{c}{R} = -0.101 + 2.27\left(\frac{r}{R}\right) - 7.14\left(\frac{r}{R}\right)^2 + 8.38\left(\frac{r}{R}\right)^3 - 3.37\left(\frac{r}{R}\right)^4 \quad (3.4)$$

With the general chord shape set, and scaled for based on radius, it can now be adjusted to suit the thrust needs of the propeller, using a linear scaling factor. This, in addition to the radius and pitch angle of the propeller, will be selected using an optimization method detailed in the next chapter.

CHAPTER 4

OPTIMIZATION TECHNIQUES

Several design parameters still need to be selected, namely radius, chord length, and pitch angle. Without greater specifics for the mission, such as payload and desired range, it is difficult to determine how to set these features to specific values. There is also the issue of trying to balance both the thrust of the propeller and its overall efficiency. In order to tackle both of these issues, an optimization method was adopted to best select these values based on the above parameters. As there is not a specific mission in mind, three propeller designs will be used, one with optimal thrust, one with optimal efficiency, and one that balances the two.

A few baseline assumptions will have to be made before this can begin. These assumptions are all based on the current MARS2020 UAV design, as it is the most complete rotor design for this atmosphere that currently exists. From online pictures, the UAV appears to have a propeller diameter of around 3ft and a wide chord that looks to be roughly 0.3r in terms of length. It has also been stated in press releases that the propeller operates at almost 3000 RPM, which will be used as the assumed RPM for this design (Northon 2018). Using these assumptions, optimization can begin.

4.1. Non-Dominated Sorting Genetic Algorithm Method of Optimization

Before a finalized method of optimization was chosen, a few other ideas were tested. These served as a basis that would eventually lead to the final method. The first method used the BEMT program to create a broad swath of data, that was then narrowed down based on the maximum Mach number limit, and an arbitrary minimum thrust limit, that was also derived from the MARS2020 UAV. This method proved to be both inefficient at generating a design, requiring constant retooling from generation to generation, but also ran into the issue of the thrust limit being completely arbitrary without a set design.

A revised method was then used, where several best fit lines of the radius vs. thrust were created, using a new RPM for each line. From these best fit lines, new data points could be plotted, showing both the Mach limit and the arbitrary thrust limit. This served to create a design space, making it superior to the previous method, but still failed in other ways. It was extremely inefficient, needing a new formula every time the chord length or pitch angle changed. Further, the thrust limit continued to be arbitrary, and the method failed to account for the efficiency of the design.

From this point it became clear that a different optimization method was required, specifically one that could optimize several variables for a minimum of two objective functions, thrust and efficiency. In addition to this, the arbitrary thrust limit needed to be dropped as it wouldn't be applicable to even slightly different designs. The method selected for this purpose was a Non-Dominated Sorting Genetic Algorithm (NSGA), a technique developed in 1994, as a means of optimizing several variables for multiple objective functions (Srinivas and Deb 1994).

NSGAs work similar to all other types of genetic algorithms. Before it is run, the user must specify a population size, a number of generations, a range that the population can fall into, and a few other variables such as crossover and mutation rates (Srinivas and Deb 1994). The workflow goes as follows. An initial population of design variables is randomly generated, based on limits set at the beginning. From here, each population member is run through the objective functions to find their values. With each of the populations variables and objective functions set, sorting on the basis of non-domination can begin. Non-domination is defined as all of a population member's objective function values being superior to the values of other population members, superior in this instance meaning smaller values. The program compares every single population member's objective function values to the others, keeping a count of all times that population member is dominated by another. After this comparison takes place, the members are ranked based on their domination count, with the lowest

counts given the highest rank. Ideally, there will be several members who go completely undominated, and earn a rank of 1.

From here, generation of the next population can begin. This comes in two forms, reproduction and mutation. For reproduction, a random selection of population members occurs, with a bias towards higher ranking members. Once two members are selected, their design variables are then averaged together to create a new population member. This occurs at a rate set by the user at the beginning of the algorithm. Similar to this, mutations also select random members of the population, again with a bias towards higher ranks. This, however, only selects one member of the population and slightly alters its design variables, with the method left up to the designer. The offspring and mutants generated through these methods then make up the next generation of the population, with a bias towards superior design choices. The process then repeats again, for as many generations as specified by the user. In theory, as the generations progress and more optimized designs are selected for reproduction and mutation, the algorithm will map out the Pareto Frontier, a range of solutions between the optimal points for all the objective functions. This range is where the best design solutions lie.

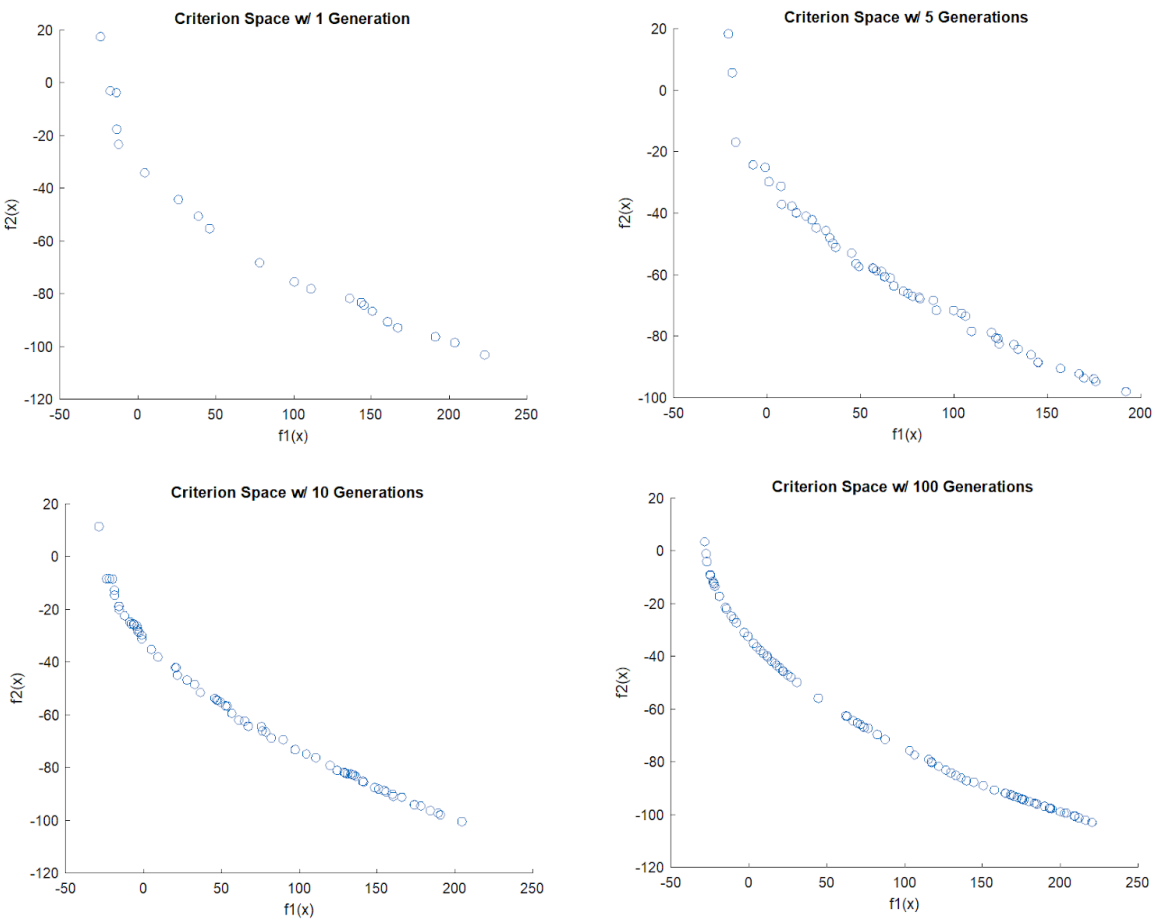

Figure 4.1: NSGA Pareto Frontier creation over several generations using a simplified example

There are multiple iterations of the basic NSGA, with a notable example being NSGA-II (Deb et al. 2000). This version of the program implemented several improvements on the original algorithm that were then adopted when creating the NSGA for the propeller. The most notable improvements were the implementation of elitism and a crowd distancing function. Elitism, in terms of genetic algorithms, is defined as the ability for members of a previous population group to carry over into the next generation of design variables. This yields multiple advantages. It requires less computation for reproduction, thus speeding up the algorithm's computation time, while also keeping good solutions from previous generations that might otherwise have been discarded by other NSGA's (Deb et al. 2000). This is

naturally done with a bias towards higher ranked members. Crowd distancing is a more effective replacement for sharing parameters originally used by NSGA's to ensure a diversity of solutions. Sharing parameters themselves are set by the designer as a means of controlling how much in common various population members may have. This makes diversity of solutions highly dependent on what the user has specified and can lead to the overall optimization failing to give an acceptable range of responses. Crowd distancing does away with this, by introducing a set parameter that is not subject to the designer's best guesses. It does so by calculating the distance between a population member's objective function values and the objective function values of other members, using a simple Pythagorean method. This creates a secondary criterion for selecting members for reproduction and mutation, with further distanced solutions being considered more diverse and thus more desirable (Deb et al. 2000).

It should be noted that there are a few approaches to calculating crowd distancing and that this project uses a different method then used in NSGA-II. The method outlined there calculates the distance only between a population member and its immediate neighbors, in terms of objective functions. This reduces the number of calculations a program must undertake, and thus improves the efficiency of NSGA (Deb et al. 2000). The method used for this project calculates the crowding distance of a member with respect to all other members of the population. Naturally this requires far more calculations per generation then the NSGA-II method, but is used for a few reasons. The first is that the objective functions used in this project are relatively simple and the computing power available to a laptop in 2020 is superior to most computers from 2000, making the issue of processing time negligible. This method is also simpler to code, as it doesn't require exceptions on either end of the Pareto Frontier, meaning less overall code. Finally, the first method may be biased against fringe solutions with one close neighbor as opposed to centralized solutions with moderately distanced neighbors, meaning overall diversity of solutions is better maintained.

The positives of this method, and rationale for its design, lie in how it can balance multiple objective functions (Srinivas and Deb 1994). Typical optimization methods for multiple objective functions require the user to create a function that combines all objective functions and weighs them by importance. The combined function is then solved using the specified method. This is limiting in multiple ways. It requires the designer to assign weight to the objective functions when there is likely no objective method for weighing their importance. Additionally, it will only provide one solution meaning that if others are desired, or a new weighting scheme is tried, the method has to be completely redone. By providing several solutions across the entire Pareto frontier, NSGA's avoid this problem and offer designers flexibility in how they select a design from the optimization. Further, it also allows for easy visualization of the criterion space when working with two objective functions, which is ideal for this project.

There are a few drawbacks that need to be mentioned, however. First is that because genetic algorithms are ultimately random, there is always the possibility that an iteration may fail to produce a viable Pareto Frontier. There is very little chance that this outcome occurs, and thanks to the simple nature of the problem being optimized, it is very quick to just run another round of optimization, making this issue negligible. In addition to the chance of failure, there is also the algorithm's inherent bias towards middling solutions. Because it is strongly predisposed towards choosing non-dominated solutions, which are more likely to occur at the center of the Pareto Frontier, the algorithm may neglect solutions closer to the fringes and fail to give a full scope of the criterion space. However, from the results given from this experiment it should be easy to see what trends emerge between the design variables and the objective functions, making the extreme solutions simple to infer.

As this method seems uniquely positioned to give both the results necessary for this design, and because the drawbacks to using it are so negligible, the NSGA method of optimization was adopted for the final design of this project.

4.2. Model Generation

Before optimization could take place, objective functions for thrust and efficiency needed to be generated, using radius, chord length, and pitch angle as the design variables. While the full BEMT algorithm could have been used for the objective functions, it is already a large algorithm and the computation time for the NSGA would likely have risen sharply if this was the case. There is also an issue with the BEMT analysis occasionally returning unrealistic values, which is discussed further below. Simplified models reduce this computation time, allow for rapid optimization, and can account for the outliers.

The models were made using non-linear regression, similar to the coefficient correction done in section 2. In order to generate this model, the BEMT program needed to be run, covering a large swath of data of various radius, chord, and pitch values. In this instance values of Radius = 0.75-1.75 ft, Pitch = 5 - 25^{o}, and Chord = 2.4-2.8x Equation 3.4, were used. All of these values were determined based on the MARS2020 UAV and the currently established dimensions, as outlined at the beginning of this chapter. The radius was restricted to 1.75 ft, as that is the radius where the propeller will exceed a maximum Mach Number of 0.7, assuming RPM = 3000. Pitch angle was restricted to 5 - 25^{o} due to the unpredictability of the induced airflow angle, which could cause a sudden shift in airflow depending on the angle. It is also unclear what exactly the lift curve of the S1223Mod3 may look like post-stall, and thus it is best to avoid this range. The chord observed in the MARS2020 UAV has a linear decrease as it moves down the radius of the blade, with an estimated maximum value of around 0.25-0.3R. The current chord formula, seen in equation 3.4 would see a significant decrease in chord lengths near the tip compared to this linear model. As such, a maximum chord of 0.4R will be allowed, as this will quickly decrease near the tips and keep torque and power from becoming excessive. A multiplicative factor, called here the Chord Factor (CF), of 2.4-2.8 will be applied to Equation 3.4 to create this result.

Once these data points were generated, they needed to be scanned for outliers and then fit to a pre-made model. Due to the recursive nature of the induced airspeed calculation, on occasion the BEMT program can return either very large or very small numbers, which are out of step with the rest of the data. Using MATLAB's built in functions, outliers were identified and removed if they were more than three median average deviations from the median of the data ("Find Outliers in Data - MATLAB Isoutlier" n.d.). Both thrust and efficiency were checked for outliers, and data was grouped by constant radius values, as this variable had the greatest effect on the range of values.

Before the data could be fit, a model needed to be established. Similar to the non-linear regression in chapter two, a few different configurations were attempted before coming to a final set up. The model used for both thrust and efficiency is:

$$T/Eff = R^{\beta_1} \cdot CF^{\beta_2} \cdot (\beta_3 + \beta_4\varphi + \beta_5\varphi^2 + \beta_6\varphi^3 + \beta_7\varphi^4 + \beta_8\varphi^5 + \beta_9\varphi^6) \quad (4.1)$$

The rationale for the model is as follows. Thrust and Power are both heavily dependent on lift and drag. These values are directly proportional to radius, chord, and the lift and drag coefficients. As the relationship is almost directly proportional for radius and chord, it is sensible to keep it the same for the model, with exponential correction factors to account for the missing variables. As for accounting for lift and drag coefficients, these can be modeled using a polynomial relationship with their AoA, and as such a polynomial relationship makes sense for the pitch angle. When the data is fit to this model it yields the following results.

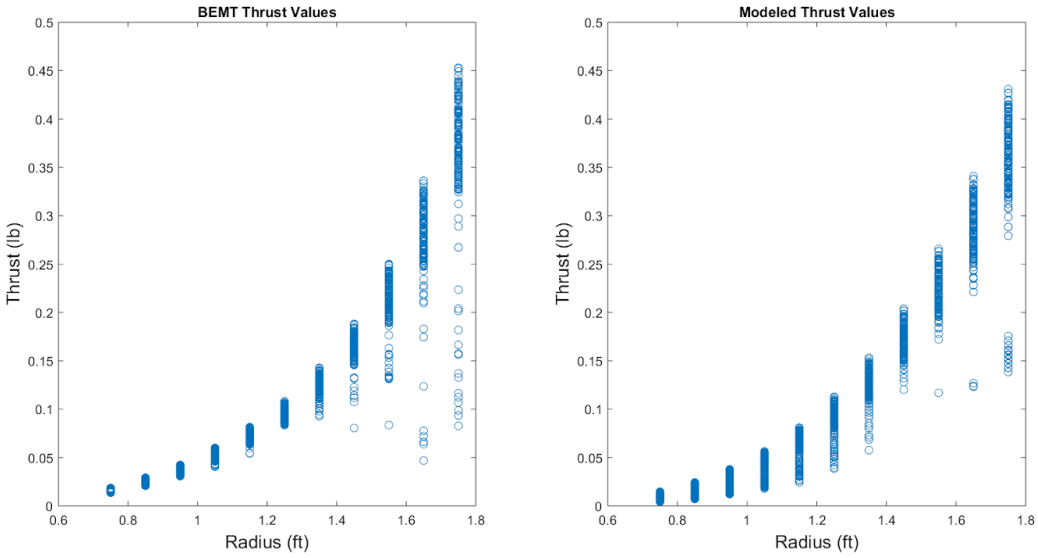

Figure 4.2: BEMT Data vs. The Nonlinear Regression Model for Thrust

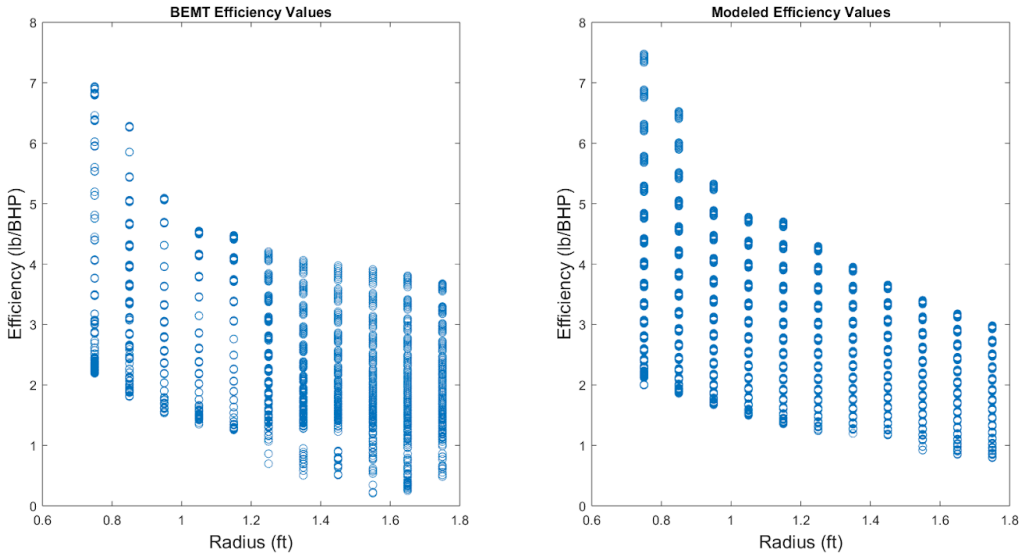

Figure 4.3: BEMT Data vs. The Nonlinear Regression Model for Efficiency

Constant	Thrust Model Value	Efficiency Model Value
β_1	3.9834	-1.1491
β_2	1.5745	0.2356
β_3	-0.0026	4.8263
β_4	0.0062	0.3168
β_5	-0.0012	-0.1584
β_6	1.1×10^{-4}	0.0189
β_7	-5.0×10^{-6}	-0.0011
β_8	9.9×10^{-8}	3.5×10^{-5}
β_9	-5.9×10^{-10}	-4.3×10^{-7}

Table 4.1: Model Coefficients for Thrust and Efficiency

Generally speaking, all of these relationships make sense. We see both that higher radii tend to lead to much higher thrust, but are inversely proportional to the efficiency of the propeller. This makes sense as efficiency in this case is just thrust/power, and power is calculated similar to thrust, but with an extra R term. Chord also has a slightly higher then linear effect on the thrust, and a considerably smaller effect on the efficiency, which makes sense when considering the thrust and power equations contain the same number of chord terms. The constants used for the pitch angle all generally follow the pattern seen in the AoA vs. Cl and Cd curves for various airfoils, which is expected. As can be seen in the graphs, the model does a good job of replicating the BEMT results, although it seems to have a smaller range of thrust and efficiency values when the radius is low and a smaller range when radius is high. It also has a difficult time replicating certain stray values, particularly at higher radiuses. This is likely due to the inability of the model to recreate the recursive nature of the induced airflow calculation, which has a

small, but noticeable effect on the final BEMT values. With the model generated, and close to the BEMT results, it can now be placed in the NSGA and optimization can commence.

4.3. Optimization Results

The parameters of the optimization were a population size of 200, 100 generations, a 50% crossover rate for reproduction, and a 30% mutation rate. As seen in Figure 23, 100 generations is plenty of time to shape the Pareto frontier of relatively simple problems, and 200 population members give a wide array of answers while still computing extremely quickly. Both the crossover and mutation rates are fairly standard for this type of genetic algorithm, introducing a sufficient number of new values, without crowding out older answers. Below is an example of what the graphed Pareto Frontier looks like, with tables indicating typical answers for high-thrust, high-efficiency, and compromise designs.

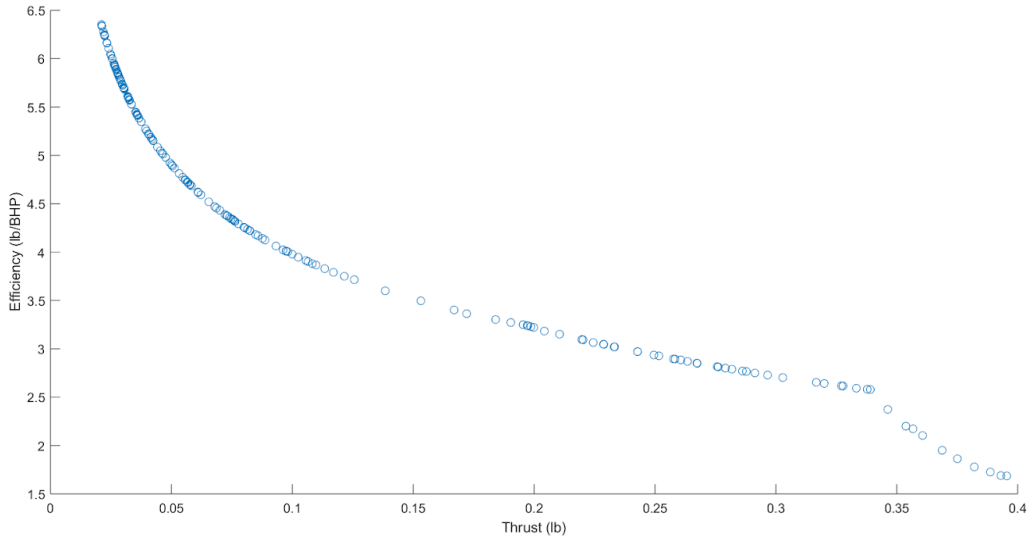

Figure 4.4: Example Pareto Frontier of the Optimization

Radius (ft)	Radius (m)	CF	Pitch Angle (deg)	Thrust (lb)	Thrust (N)	Efficiency (lb/BHP)	Efficiency (N/W)
1.7412	0.5307	2.7874	13.9855	0.4011	1.7842	1.2929	0.0077
1.7487	0.5330	2.7114	10.6251	0.3887	1.7290	1.7258	0.0103
1.7366	0.5293	2.6205	7.3061	0.3710	1.6503	2.3144	0.0138
1.7197	0.5242	2.7891	12.6577	0.3822	1.7001	1.4773	0.0088
1.7292	0.5271	2.5999	7.0252	0.3618	1.6094	2.3786	0.0142

Table 4.2: Example Optimization Data for High Thrust Propeller Design

Radius (ft)	Radius (m)	CF	Pitch Angle (deg)	Thrust (lb)	Thrust (N)	Efficiency (lb/BHP)	Efficiency (N/W)
0.7821	0.2384	2.7357	5.3689	0.0168	0.0747	6.8720	0.0410
0.7828	0.2386	2.7385	5.3741	0.0168	0.0747	6.8632	0.0409
0.7588	0.2313	2.6463	6.5234	0.0141	0.0627	6.4240	0.0383
0.8270	0.2521	2.7435	5.5603	0.0211	0.0939	6.3524	0.0379
0.8390	0.2557	2.5818	5.9710	0.0203	0.0903	5.9583	0.0355

Table 4.3: Example Optimization Data for High Efficiency Propeller Design

Radius (ft)	Radius (m)	CF	Pitch Angle (deg)	Thrust (lb)	Thrust (N)	Efficiency (lb/BHP)	Efficiency (N/W)
1.4186	0.4324	2.6988	5.4684	0.1760	0.7829	3.4285	0.0205
1.3746	0.4190	2.6740	5.6503	0.1532	0.6815	3.4961	0.0209
1.3521	0.4121	2.6558	5.2718	0.1414	0.6290	3.6654	0.0219
1.3183	0.4018	2.7208	5.5137	0.1331	0.5921	3.7236	0.0222
1.2967	0.3952	2.6434	5.2399	0.1187	0.5280	3.8512	0.0230

Table 4.4: Example Optimization Data for Compromise Propeller Design

Multiple trends are apparent in this data, reflecting how changing the radius, chord length, and pitch angle may affect the overall propeller performance. The strongest effect clearly comes from the radius of the propeller, which was already apparent in model generation. All of the high thrust designs have radii close to the 1.75 ft limit, the high efficiency designs all are close to the 0.75 ft limit, and the compromise designs all fall in the middle. This trend is easy to observe, although the slight differences in these values do help indicate what relationship CF and pitch might have with overall performance.

The next relationship worth exploring is the CF, which seems to bias towards higher values in all three designs, never dipping below halfway point of 2.6. This result makes sense. In both the thrust and efficiency model, the CF factor had a much smaller exponent then the radius factor, with the exponent in the efficiency model being close to 0. In general, the high thrust values seem to favor higher CF values, and high efficiency seems to value the opposite, but again, this is a small effect. In this instance, because its effect on efficiency is so small and it does positively contribute to higher thrust, CF can be kept at a higher value. There are drawbacks to higher chord length designs, dealing with stability, balance, and wake, but this project will not be exploring these factors.

The most interesting factor involved here is the pitch angle. Typical Pareto Frontiers look like rounded corners, that then stretch back in straight lines. This frontier, by contrast, has a notable kink to it, steeply changing direction at higher thrust values. The cause of this might not be readily apparent, but likely has to do with how the model replicates the airfoil's loss of efficiency at higher AOA's. The modified S1223 airfoil experiences a sharp decline in its C_l/C_d ratio after peaking, typically around the 3-10^o range. In the vast majority of NSGA run throughs performed, the majority of pitch angles are found to be in the 5-6^o range, where efficiency is typically the highest. The reason for this is likely the selection bias of the algorithm. Designs with higher pitch angles saw large drops in their efficiency, which likely lead to an inflation of their domination count. This means that only designs with the highest of thrusts would be able to survive, making higher pitch angles relegated to that corner of the Pareto Frontier.

This reality is reflected when looking at the sample optimization data above. For both the compromise and high-efficiency designs the pitch angle never exceeded 6.6°, whereas for the high thrust designs they all lie between 7-14°. These values line up roughly with both the maximum C_l/C_d value and the C_{lMAX} value of the airfoil, when accounting for the shift with induced velocity. This uncertainty due to the induced velocity makes it important to see the precise effect of shifting the high thrust propellers angle within the 7-14° range, as this could lead to either improvements or reductions in performance. For this analysis, the most helpful examples are the top two designs in Table 4.2, and the bottom two designs. In both these comparisons, the radius and CF of the designs are roughly the same, but the pitch angles are clearly different, making them the deciding factor in the propeller performance. For the top two designs, we see that the lower pitch angle yields both a slightly higher thrust and a considerably higher efficiency. Similarly, the bottom two designs show the lower pitch angle having a barely lower thrust, but a considerably higher efficiency, although there is a more noticeable difference in its R and CF values. This seems to indicate that raising the pitch angle above 9-10° doesn't yield vastly superior thrust, but may lead to lower efficiencies.

Given all this information, it becomes easy to finalize the design of these three propellers. For the high thrust design, the radius and CF factor both need to be kept as long as possible to maximize thrust, while the angle needs to be close to the C_{lMAX} value without stalling out. As such, values of R = 1.75 ft, CF = 2.8, and $\phi = 10°$. For the high efficiency design, radius needs to be minimized and the pitch angle needs to be kept low, however, the CF value can remain fairly large without any detrimental effects on the overall efficiency. Final dimensions for this design will be R = 0.75 ft, CF = 2.8, and $\phi = 5.5°$. Finally, for the compromise design, middle ground needs to be selected. Radius will be kept at a middling value between the design extremes. CF can be kept high, as this has only a positive effect on the thrust, and minimal effect on the efficiency. The pitch angle will need to be kept low, as the gains in thrust don't

seem to be able to counteract the large drops in efficiency that occur with this design. For this design R = 1.33 ft, CF = 2.8, and $\phi = 6.0^{o}$.

Design Type	Radius (ft)	CF	Pitch Angle (deg)
High Thrust	1.75	2.8	10
High Efficiency	0.75	2.8	5.5
Compromise	1.33	2.8	6.0

Table 4.5: Final Dimensions for the Three Propeller Designs

With the final designs for the propellers set, the modeling and testing process for the propellers can begin. The effectiveness of these designs, and the observations of the optimization process will be reexamined after testing data has been gathered.

CHAPTER 5

MODELING AND TESTING

With theoretical propellers designed and BEMT estimates made, the next step is to verify the estimates. Traditionally, verification would be done using physical testing, likely using 3D printed propellers, a vacuum chamber, and load cells. However, due to both available hardware limitations, and current global events, physical testing was unachievable in a realistic time frame. As such, a Computational Fluid Dynamics (CFD) simulation using ANSYS software was selected as the verification method. This option was cheaper and allowed the work to be done remotely. While physical test results would be preferable, ANSYS Fluent software has been shown to give accurate thrust and power estimates for propellers at Re < 100,000, giving results with less than 5% error when compared to physical testing (Kutty and Rajendran 2017).

5.1. Modeling Procedure

Before a simulation of propeller performance can be done, the geometry of the propeller must be created. Initially geometry was modeled using SolidWorks, a popular CAD program. However, there were compatibility issues with using SolidWorks files with ANSYS Fluent, and thus work was moved to two dedicated CAD programs, SpaceClaim and DesignModeler, both part of the ANSYS package. The former was used to create the geometry of the propeller while the latter was used to generate the flow domains surrounding it.

With multiple propellers needing to be modeled for this project, a standardized method for modeling was employed. The raw airfoil coordinates for the modified S1223 foil were taken from the XFLR5 file. A simplified version of these coordinates was used, transforming the number of coordinates from 81 down to 36. The reason for this was that simplifying the number of surfaces along the propeller greatly improved the simulation performance while in ANSYS. Next, using a modified version of BEMT, these coordinates were transformed to match the dimensions of the propeller blade. This was done

using simple matrix transformations to stretch, rotate, and translate the airfoil coordinates to match the various blade stations. While the initial BEMT formula did not take into account any forward translation of the blade, this was done in the 3D model to ensure the blade was centered with the hub. This transformation would have no effect on the BEMT results.

Once the coordinates were generated, they were placed into .txt files which could then be imported into SpaceClaim (“Importing and Exporting” n.d.) and then blended into a single blade. Initial attempts to do this were unsuccessful due to the narrow and sharp angles at the trailing edge of the airfoil. To create a geometry which would work with ANSYS, a small portion of the trailing edge was cut off, shortening the overall chord and reducing the overall camber. The result of this was a less accurate representation of the propellers geometry, but a representation that would work with the simulation program. The overall effects of this simplification are discussed further in the results section.

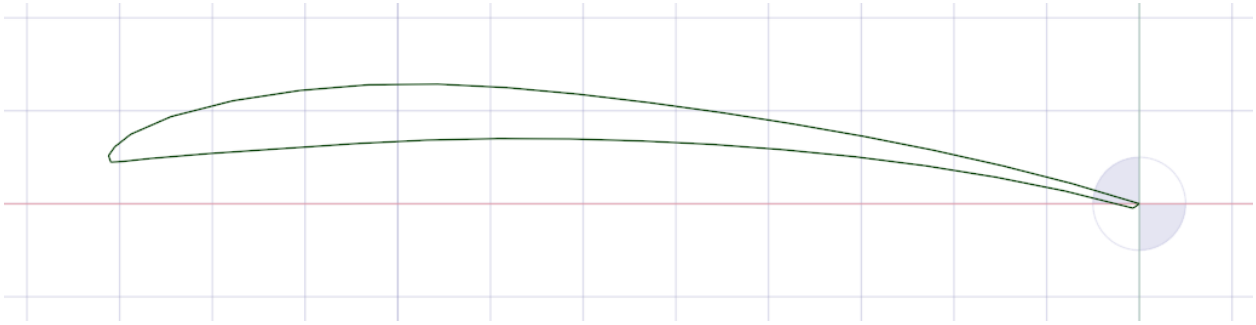

Figure 5.1: Airfoil Cross Section, with Shortened Trailing Edge

Once the full blade is created using the blend tool, the blade can then be copied and rotated about the y axis to create both propeller blades. As a final step, the hub is created using a circle tool on the origin, and combined with the propellers to create a single, solid part. The radius of the hub is set so that it will encompass the entirety of the nearest blade station. Chamfers are also added to the hub, to better reflect the shape of an actual propeller hub.

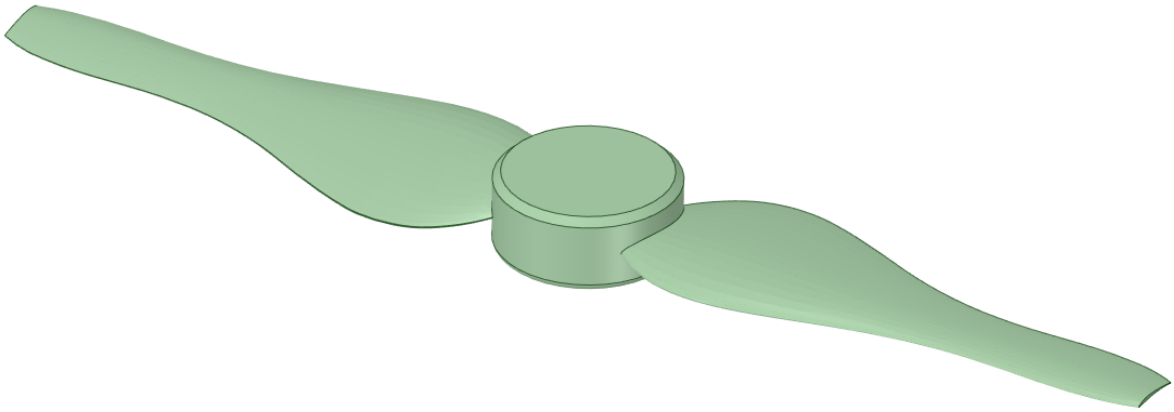

Figure 5.2: Example of a Fully Rendered Propeller Model

Once the final blade geometry was created, the work was moved to the Design Modeler program. In here, two cylindrical enclosures are created surrounding the propeller. These are to act as the rotational and static domains for the simulation. The smaller, rotational enclosure is set to have boundaries of 0.05R beyond the blades surface, radially and along the vertical axis. The boundaries of the larger, static enclosure are set to 5R in the same directions. The rotational domain is kept smaller to only encompass the blades immediate geometry, while the static domain is large enough to prevent significant wall effects from influencing the data. Both of these specifications are made to exceed the criteria of previous studies (Kutty and Rajendran 2017). A boolean is then created to subtract the propeller geometry from the rotating domain, creating a shell within the domain that will act as a solid wall during the simulation. The rotating domain is then subtracted from the static domain in another boolean, to prevent any geometry overlap. With the geometry set, meshing and simulation can begin.

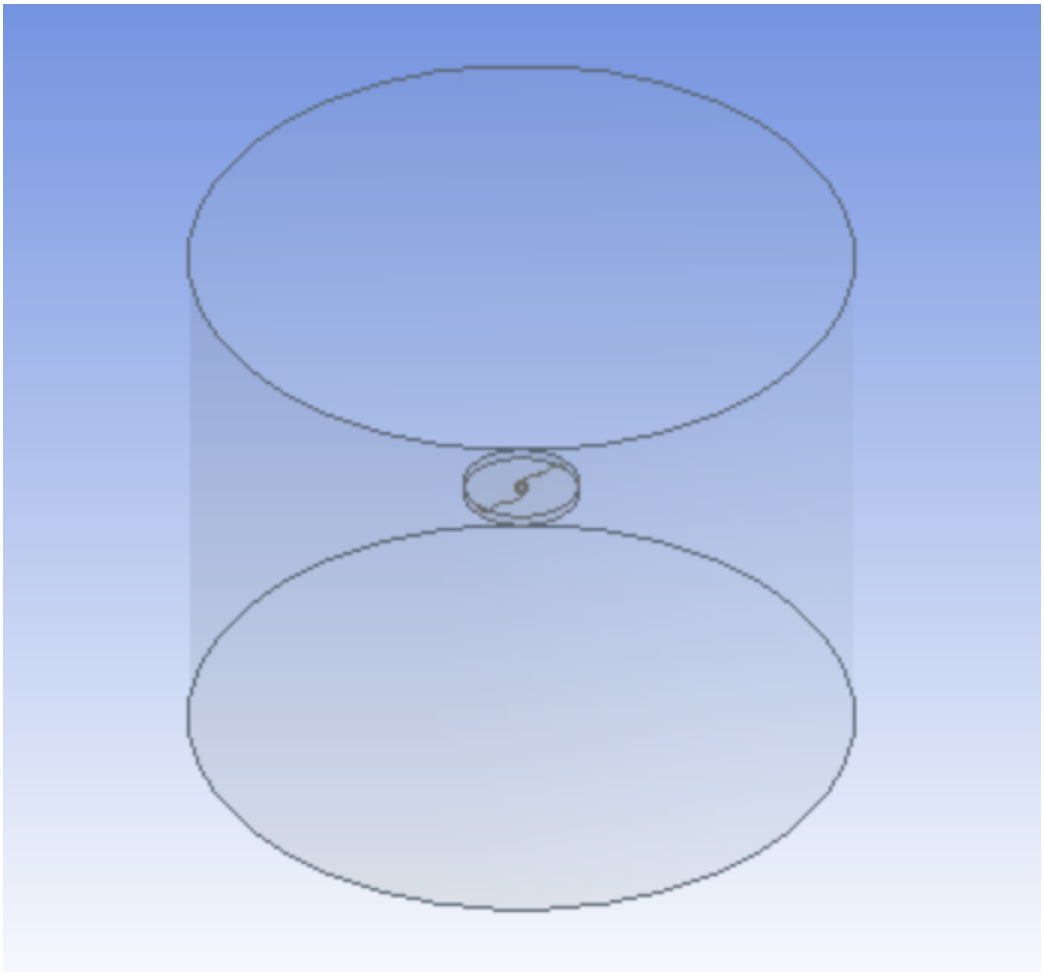

Figure 5.3: Full Simulation Geometry

5.2. ANSYS Analysis Parameters

Meshing of the geometry is done using the built in Meshing Software for ANSYS. The purpose of meshing is to sub-divide the geometry into several smaller elements where the CFD equations can be performed. As a general rule of thumb, high element counts lead to more accurate results, but take significantly longer to process. To ensure the accuracy of results, a 'Mesh Independence Study' is undertaken, where multiple meshes are used for the same simulation, to ensure the results are consistent regardless of the mesh usage.

For this project standard, unstructured mesh generation with default sizing values were used, with a few key exceptions. A sizing function was applied to the propeller faces, as this is the most

detailed portion of the geometry where the most important calculations are taking place. This creates a more refined mesh in this area, and thus leads to more accurate results. The sizing function is set to ‘Capture Curvature’ to ensure necessary meshing details for the propeller curves. The ‘Min Size’ is also manually set, to ensure it is smaller than the width of the modified trailing edge. This is necessary to successfully generate a mesh. The final setting that is altered is the ‘Growth Rate’ which determines how quickly adjacent elements can grow compared to each other. Higher growth rates allow the mesh to capture the thin portion of the propeller blades, while not excessively inflating element count and slowing down the simulation. Default ‘Growth Rate is set to 1.2, while this project used values of 1.3 and 1.4. These two separate values are used for the mesh independence study, with these two rates giving similar performance results with significantly different cell counts. This process worked for all three propellers.

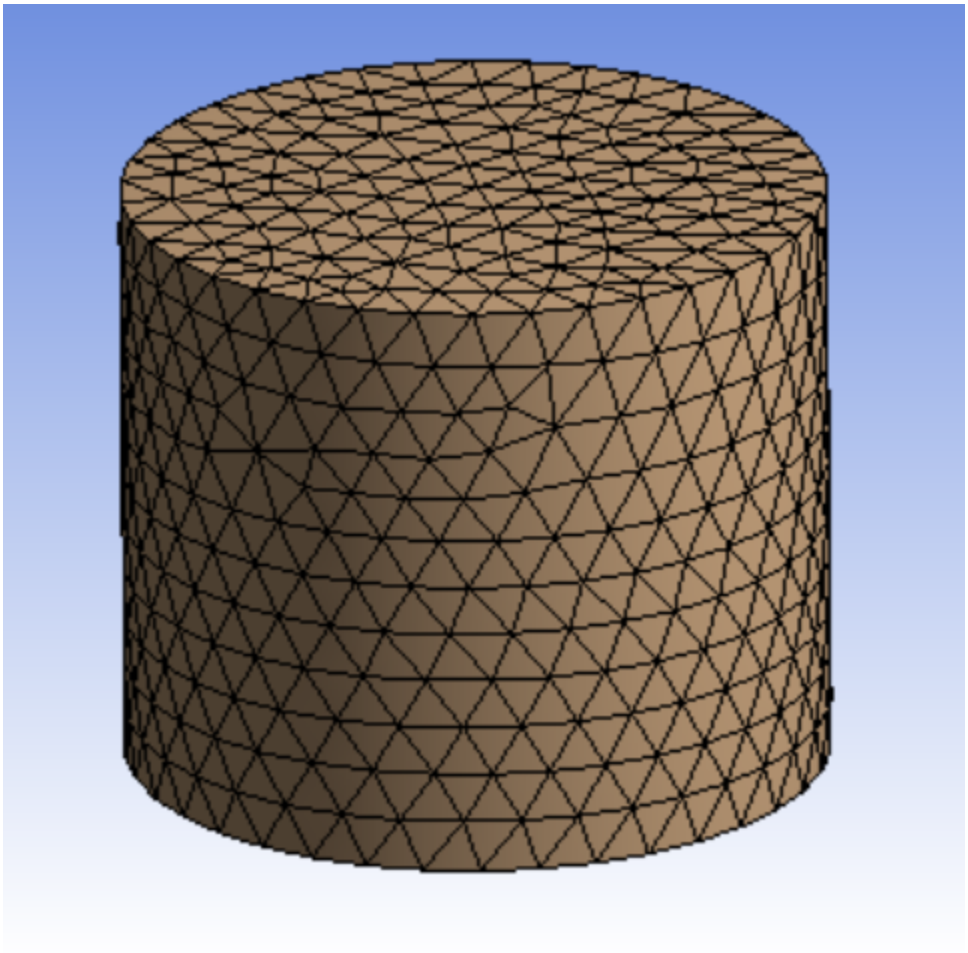

Figure 5.4: Outside View of the Meshing Used

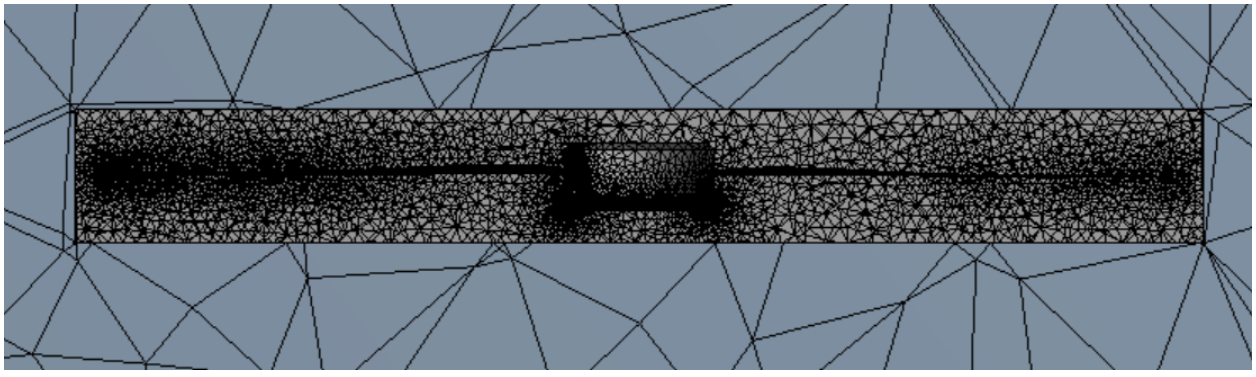

Figure 5.5: Cross Section view of Mesh, showing difference in detail between the propeller and other the outside mesh

Design Type	Mesh Growth Rate	Nodes	Elements
High Thrust	1.3	327,387	1,755,259
	1.4	217,793	1,135,740
High Efficiency	1.3	255,203	1,375,987
	1.4	170,990	898,975
Compromise	1.3	257,441	1,379,279
	1.4	172,652	900,552

Table 5.1: Mesh Statistics for all Propellers and Simulations

From here the mesh can be imported into ANSYS Fluent software, and the parameters of the simulation can be set. This simulation is meant to test each propeller in a Martian atmosphere, while rotating at 3000 RPM. Atmospheric conditions are derived from the equations in section 2, with P = 13.6 lbf/in^2, $\rho = 9.07308x10^{-4}$ lbf/in^3, and $\mu = 7.94699x10^{-6}$ lbm/ft*s. Frame motion is used to have the rotational domain rotate in reference to the static domain. This enables and accounts for the rotation of the propeller within the rotating domain, as well. The most significant boundary conditions are the propeller faces and sides of the static domain being treated as solid walls, while the top and bottom of the static domain are pressure outlets. This allows fluid flow over and around the blade, with minimal interference. Neither pressure outlet is given a velocity, as the simulation is meant to look at hover conditions.

The analysis being performed is a pressure based, transient solution, allowing the conditions to change with the propeller motion, and eventually converge on a solution. For these simulations, the default Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) are used, as they were found sufficient for other low Reynolds number analysis (Kutty and Rajendran 2017). There is more contention with regard to the appropriate turbulence model, as there are several options with different advantages. Kutty and Rajendran 2017, on which this simulation has been based, found that the standard k-ω model

gave the most accurate results (Kutty and Rajendran 2017). However, their study occurs at higher Reynolds numbers than this simulation, and fails to mention whether Low Re corrections were used. Nonetheless, the SST k-ω model was adopted, as it has shown great accuracy when dealing with adverse pressure gradients, which are likely to occur within this low Re flow (Argyropoulos and Markatos 2015). Low Re corrections were used along with this model.

5.3. Results and Comparison to BEMT

All simulations finished without any convergence issues. Final performance values were also similar across different meshes, fulfilling the mesh independence requirement. Results of the different simulations can be seen below.

Design Type	Mesh Growth Rate	Thrust (lbf)	Moment (lbf*ft)	Efficiency (lbf/BHP)
High Thrust	1.3	0.5387	0.1684	5.6008
	1.4	0.5376	0.1684	5.5880
High Efficiency	1.3	0.0058	0.0025	4.1137
	1.4	0.0059	0.0025	4.2250
Compromise	1.3	0.0930	0.0334	4.8739
	1.4	0.0903	0.0332	4.7528

Table 5.2: Simulation Results for all Propellers and Meshes

Design Type	Mesh Growth Rate	Thrust (N)	Moment (N*m)	Efficiency (N/W)
High Thrust	1.3	2.3962	2.4574	0.0334
	1.4	2.3916	2.4582	0.0333
High Efficiency	1.3	0.0259	0.0362	0.0245
	1.4	0.0264	0.0359	0.0252
Compromise	1.3	0.4136	0.4874	0.0291
	1.4	0.4015	0.4852	0.0284

Table 5.3: Simulation Results for all Propellers and Meshes (SI conversion)

With the agreement in numbers between meshes, these values can then be averaged and compared to the BEMT simulation numbers.

	High Thrust Propeller	High Efficiency Propeller	Compromise Propeller
BEMT Thrust (lbf)	0.4360	0.0027	0.1363
BEMT Thrust (N)	1.9394	0.0120	0.6063
ANSYS Avg. Thrust (lbf)	0.5382	0.0059	0.0916
ANSYS Avg. Thrust (N)	2.3940	0.0262	0.4075
Percent Error (%)	-18.98	-54.08	48.78
BEMT Moment (lbf*ft)	0.3688	0.0047	0.0676
BEMT Moment (N*m)	5.3822	0.0686	0.9865
ANSYS Avg. Moment (lbf*ft)	0.1684	0.0025	0.0333
ANSYS Avg. Moment (N*m)	2.4576	0.0365	0.4860
Percent Error (%)	118.0	90.35	102.9
BEMT Efficiency (lbf/BHP)	2.070	6.717	3.533
BEMT Efficiency (N/W)	0.0123	0.0401	0.0211
ANSYS Avg. Efficiency (lbf/BHP)	5.594	4.169	4.813
ANSYS Avg. Efficiency (N/W)	0.0334	0.0249	0.0287
Percent Error (%)	-62.00	61.10	-26.59

Table 5.4: Simulation Results Compared to BEMT Estimates

From this data, a few trends emerge. First is that there is far more agreement between the thrust results of ANSYS and BEMT, than between their results for moment. This lack of agreement also continues into the efficiency results, as this is based on both the thrust and moment. Further, while the BEMT results predict a trend where the efficiency of the propeller improves with the decrease in radius, ANSYS results give the opposite result. There are several possible reasons for all of these results, mostly relating to the accuracy of the BEMT methods and the nature of the ANSYS simulation.

The first thing that can be examined is the results for thrust. This had the best agreement of all the tested values, with its largest percent error only being -54.08% and an average absolute error of 40.61%. What is perhaps most noticeable about these results is that the absolute error seems to decrease as the radius of the propeller increases. This increase in radius also leads to an increase in the Reynolds numbers at all blade stations, by increasing both the chord length and radial velocity. As was examined in section 2 and 3,many of the data and corrections on which BEMT is built are generally less accurate at lower Reynolds numbers. This includes both the XFOIL polar data, as well as the custom correction put in place to fix that issue (Chen and Bernal 2008; Maughmer and Coder 2010; Miley 1982). With this in mind, it seems natural that the largest propeller, and thus the one that experiences the largest Reynolds numbers, would give the most accurate results.

While the thrust results proved to be generally accurate, the moment results had far larger issues. Here the BEMT estimates are all significantly larger than the simulation results, with an average error of 103.75%. There are several possible reasons for this, although the extent to which they effect the results is unclear. First, as mentioned in the previous paragraph, there is the lack of data surrounding BEMT at extremely low Re, and thus accuracy issues can derive from this. Moment is most heavily influenced by the drag force, and thus any error is likely to stem from here. As mentioned in sections 2.2 and 2.5, no compression correction was used due to the inability to distinguish between skin friction and pressure drag under certain conditions and no custom correction could be made for C_d due to a lack of distinguishable trends in the experimental data. We also see a similar trend in section 2.6 while recreating a separate experiment (Shrestha et al. 2016). This showed the BEMT estimates for power (which is proportional to moment) being roughly double the experimental power values, which is consistent with these new results. Both these results point to the drag polars derived from XFOIL and used in the BEMT algorithm as being far higher than the actual values. Another possible explanation for these results is the simplifications made during the simulation. By cutting off the trailing edge of the

propeller, the effective camber of the propeller is greatly reduced. Previous studies found a heavy link between high camber and overall drag of the airfoil at low Re (Shyy et al. 2007; Sunada et al. 2002). Thus, by reducing the overall camber of the airfoil, drag may have been reduced, and moment may have also been reduced as a result. The extent to which it may have been reduced is hard to quantify, however.

The final major result to examine is efficiency. The overall error between the BEMT and ANSYS numbers is a combination of the previous two metrics, and is likely a result of the previously discussed issues. The noticeable trend here is that the ANSYS values had the opposite trend of the BEMT results, with the High Efficiency model having the lowest efficiency, and the High Thrust having the highest efficiency. There are several possible reasons for this. Chief amongst them relates back to the increase in Reynolds number with the larger propellers. Lower Reynolds numbers generally trend towards lower C_l/C_d ratios, thus one would expect the larger propellers with the larger Re values to be more efficient. This could be enough to offset the additional radius term in the power calculation, which led to the trend seen in the BEMT results. How this relates to the previously stated sources of error is difficult to say, however.

CHAPTER 6

CONCLUSIONS AND FUTURE PLANS

With the results of this initial design testing finished, conclusions can be stated and future plans can be drawn up. The chief conclusion of this testing shows that a UAV on Mars is completely feasible, if made light enough. Propeller designs were found capable of producing over 0.5 lbf of force while hovering. Further knowledge of mission parameters and power sources would be necessary to design a full drone, although these results show what some of the weight and power requirements may be. For a comparison, the Ingenuity UAV being tested by NASA on Mars will weigh just under 4 lbf, and use rotations between 2,400-3,000 RPM (Northon 2018; Greicius 2020). Factoring in the dual blade design, and the lower gravity of Mars, this design would need to generate roughly 0.75 lbf of thrust to achieve flight. While the overall radius and airfoil design of the UAV are unknown, this puts the largest tested design in the same ballpark.

Aside from this broad conclusion, these successful trials open up to a world of possibilities, both in trying new designs and improving on the procedure of this project. The designs looked at in this experiment primarily focused on balancing both thrust and efficiency. Other possible designs could focus on either of these two aspects, and also explore different types of design. This includes new airfoil designs, different chord distributions, different twists and forward speeds, and different blade numbers. With the lack of concrete data for the effects of low Reynolds number flow on propellers, any one of these new design choices would provide vital data for designing future blades.

Other changes could all come in the form of procedural changes to the design and verification process. For instance, all of the propellers in this project were tested at the same RPM value. However, the two smaller designs could have been run at higher speeds without incurring any major drag penalties from higher Mach numbers. A better testing procedure, to gauge the full capabilities of these designs, would be to set them with equal tip Mach numbers. This would allow the smaller blades to generate

more thrust, while also operating at higher Reynolds numbers, and thus possibly improving efficiency. This new design approach could also be done when creating the initial optimization functions. This could be further modified depending on the forward speed of the propeller. A new maximum rotational velocity can be determined based on an estimated maximum forward airspeed. These propeller designs could then be tested at a variety of airspeeds and advanced ratios, giving a full picture of possible propeller performance. This broader view would also help establish mass estimates for the final UAV design.

Beyond changes in the parameters being tested and controlled for, the method of testing can also be altered. The two major paths to accomplish this would be more detailed CFD analysis, and actual physical testing. More detailed CFD analysis could be done by keeping the trailing edge of the propeller as intact as possible, and significantly inflating the total element count. This may require some specialized computing software to perform this analysis in a timely and economical manner, however. While ANSYS analysis has proven to be accurate from other analyses, it is unclear to what degree this added detail will increase the accuracy of the estimate (Kutty and Rajendran 2017). Another possible use for CFD analysis would be a second set of aeronautics polar data, which can be compared to the current XFOIL data. This would best be paired with physical testing, however, to remove the chance of CFD software simply agreeing with itself.

Physical testing is preferred as it is the most likely to reflect reality. Testing of this sort has been done previously, using a vacuum chamber, hall sensors, and load cells (Shrestha et al. 2016). The most expensive of these pieces of equipment would be the vacuum chamber, which would need to be big enough for the propeller to operate without any large wall effects. Other materials can be found cheaply, although they would need to be properly calibrated and a method for recording data would need to be established. The most obvious methods for the latter would be modifying the chamber to allow a hook up to a computer outside the testing area, or simply having all the data recorded on an SD card. Testing

forward airspeeds would also be extremely difficult without a specially designed wind tunnel capable of operating at a low density.

This initial experimentation and simulation has laid the groundwork for several future studies, which can capitalize on its initial structure and results. The BEMT algorithm has proven that it can give useful preliminary estimates for propeller performance, and when combined with optimization algorithms, can produce fully functional designs. There is an established verification procedure using ANSYS Fluent for CFD analysis, which can be further refined and even shifted to physical testing. The possibilities for future designs and testing are endless.

BIBLIOGRAPHY

"2020 Toyota Prius Exterior Specs." n.d. Accessed January 14, 2020. https://www.toyota.com/prius/features/exterior/.

Anderson, John D. 2012. *Introduction to Flight*. 7th ed. New York: McGraw Hill.

Argyropoulos, C. D., and N. C. Markatos. 2015. "Recent Advances on the Numerical Modelling of Turbulent Flows." *Applied Mathematical Modelling* 39 (2): 693–732. https://doi.org/10.1016/j.apm.2014.07.001.

Brandt, John, and Michael Selig. 2011. "Propeller Performance Data at Low Reynolds Numbers." In *49th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*. Orlando, Florida: American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2011-1255.

Canis, Bill. 2015. "Unmanned Aircraft Systems (UAS): Commercial Outlook for a New Industry." Report. UNT Digital Library. September 9, 2015. https://digital.library.unt.edu/ark:/67531/metadc770623/.

Chen, Weisheng, and Luis Bernal. 2008. "Design and Performance of Low Reynolds Number Airfoils for Solar-Powered Flight." In *46th AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2008-316.

Colozza, Anthony. 1998. "High Altitude Propeller Design and Analysis Overview." Cleveland, OH: Federal Data Systems. /paper/High-Altitude-Propeller-Design-and-Analysis-Colozza/76adc694897d2a951d9216c87ca5d125cdcc80d0.

Colozza, Anthony, Mohsen Shahinpoor, Kakkattukuzhy Isaac, and Teryn DalBello. 2005. "Solid State AIrcraft Phase II Final Report." https://doi.org/10.13140/RG.2.1.2433.2247.

Corliss, William R. 1974. *The Viking Mission to Mars*. Scientific and Technical Information Office, National Aeronautics and Space Administration.

Corrigan, John J, and John J Schillings. 1994. "EMPIRICAL MODEL FOR STALL DELAY DUE TO ROTATION." In , 15. San Francisco, CA: American Helicopter Society.

Davila, Alfonso F., and Dirk Schulze-Makuch. 2016. "The Last Possible Outposts for Life on Mars." *Astrobiology* 16 (2): 159–68. https://doi.org/10.1089/ast.2015.1380.

Deb, Kalyanmoy, Samir Agrawal, Amrit Pratap, and T. Meyarivan. 2000. "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II." In *Parallel Problem Solving from Nature PPSN VI*, edited by Marc Schoenauer, Kalyanmoy Deb, Günther Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel, 849–58. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer. https://doi.org/10.1007/3-540-45356-3_83.

Deters, Robert W., Gavin Kumar Ananda Krishnan, and Michael S. Selig. 2014. "Reynolds Number Effects on the Performance of Small-Scale Propellers." In *32nd AIAA Applied Aerodynamics Conference*. American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2014-2151.

Drela, Mark. 1989. “XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils.” In *Low Reynolds Number Aerodynamics*, edited by Thomas J. Mueller, 1–12. Lecture Notes in Engineering. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-84010-4_1.

Durand, William F., and E. P. Lesley. 1925. “Comparison of Model Propeller Tests with Airfoil Theory.” https://core.ac.uk/display/42795131.

E-CFR: TITLE 14—Aeronautics and Space. n.d. *Electronic Code of Federal Regulations*. Vol. TITLE 14—Aeronautics and Space. Accessed December 17, 2019. https://www.ecfr.gov/cgi-bin/text-idx?SID=130e90af563d4e981961aa3ce77b8119&mc=true&tpl=/ecfrbrowse/Title14/14cfr107_main_02.tpl.

Eppler, Richard. 2012. *Airfoil Design and Data*. Springer Science & Business Media.

“Find Outliers in Data - MATLAB Isoutlier.” n.d. Accessed April 7, 2020. https://www.mathworks.com/help/matlab/ref/isoutlier.html.

Froude, W. 1920. “On the Elementary Relation Between Pitch, Slip, and Propulsive Efficiency.” Report. UNT Digital Library. 1920. https://digital.library.unt.edu/ark:/67531/metadc53602/m1/2/.

Glauert, H. 1928. “The Effect of Compressibility on the Lift of an Aerofoil.” *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 118 (779): 113–19. https://doi.org/10.1098/rspa.1928.0039.

Glauert, H. (Hermann). 1948. *The Elements of Aerofoil and Airscrew Theory*. Cambridge [Eng.] University Press. http://archive.org/details/elementsofaerofo00glau.

Golombek, M. P., R. A. Cook, T. Economou, W. M. Folkner, A. F. C. Haldemann, P. H. Kallemeyn, J. M. Knudsen, et al. 1997. “Overview of the Mars Pathfinder Mission and Assessment of Landing Site Predictions.” *Science* 278 (5344): 1743–48. https://doi.org/10.1126/science.278.5344.1743.

Green, James L., and John M. Grunsfeld. 2013. “Announcement of Opportunity Mars 2020 Investigations.” NASA.

Greicius, Tony. 2015. “Curiosity Overview.” Text. NASA. January 20, 2015. http://www.nasa.gov/mission_pages/msl/overview/index.html.

———. 2020. “6 Things to Know About NASA’s Ingenuity Mars Helicopter.” Text. NASA. July 14, 2020. http://www.nasa.gov/feature/jpl/6-things-to-know-about-nasas-ingenuity-mars-helicopter.

Gudmundsson, Snorri. 2014. *General Aviation Aircraft Design: Applied Methods and Procedures*. First. Butterworth-Heinemann.

Hitchens, Frank. 2015. *Propeller Aerodynamics: The History, Aerodynamics & Operation of Aircraft Propellers*. Andrews UK Limited.

“Importing and Exporting.” n.d. Accessed August 7, 2020. http://help.spaceclaim.com/2015.0.0/en/Content/Importing_and_exporting.htm.

Keane, John F., and Stephen S. Carr. 2013. “A Brief History of Early Unmanned Aircraft.” *John Hopkins APL Techincal Digest*, 2013.

Kunz, Peter J. 2003. “Aerodynamics and Design for Ultra-Low Reynolds Number Flight.” Dissertation, Stanford University.

Kutty, Hairuniza Ahmed, and Parvathy Rajendran. 2017. “3D CFD Simulation and Experimental Validation of Small APC Slow Flyer Propeller Blade.” *Aerospace* 4 (1): 10. https://doi.org/10.3390/aerospace4010010.

Lee, Chi-Seng, Weng Pang, Sutthiphong Srigrarom, Di-Bao Wang, and Fei-Bin Hsiao. 2006. “CLASSIFICATION OF AIRFOILS BY ABNORMAL BEHAVIOR OF LIFT CURVES AT LOW REYNOLDS NUMBER.” In *24th AIAA Applied Aerodynamics Conference*. San Francisco, California: American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2006-3179.

Lindenburg, C. 2003. “Investigation into Rotor Blade Aerodynamics.” *National Renewable Energy Laboratory*, July, 114.

MacNeill, R., and D. Verstraete. 2017. “Blade Element Momentum Theory Extended to Model Low Reynolds Number Propeller Performance.” *The Aeronautical Journal* 121 (1240): 835–57. https://doi.org/10.1017/aer.2017.32.

Manwell, J. F., J. G. McGowan, and Anthony L. Rogers. 2009. *Wind Energy Explained: Theory, Design and Application*. 2nd ed. Chichester, U.K: Wiley.

mars.nasa.gov. n.d. “Summary | Rover.” NASA’s Mars Exploration Program. Accessed December 18, 2019. https://mars.jpl.nasa.gov/msl/spacecraft/rover/summary.

Masters, I, J C Chapman, M R Willis, and J a C Orme. 2011. “A Robust Blade Element Momentum Theory Model for Tidal Stream Turbines Including Tip and Hub Loss Corrections.” *Journal of Marine Engineering & Technology* 10 (1): 25–35. https://doi.org/10.1080/20464177.2011.11020241.

Maughmer, Mark D., and James G. Coder. 2010. “Comparisons of Theoretical Methods for Predicting Airfoil Aerodynamic Characteristics.” AIRFOILS INC PORT MATILDA PA. https://apps.dtic.mil/docs/citations/ADA532502.

Meola, Andrew. 2017. “Drone Market Shows Positive Outlook with Strong Industry Growth and Trends.” Business Insider. July 13, 2017. https://www.businessinsider.com/drone-industry-analysis-market-trends-growth-forecasts-2017-7.

Miley, S. J. 1982. “Catalog of Low-Reynolds-Number Airfoil Data for Wind-Turbine Applications.” RFP-3387. Rockwell International Corp., Golden, CO (USA). Rocky Flats Plant; Texas A and M Univ., College Station (USA). Dept. of Aerospace Engineering. https://doi.org/10.2172/5044823.

“Nonlinear Regression - MATLAB & Simulink.” n.d. MathWorks. Accessed February 16, 2020. https://www.mathworks.com/help/stats/nonlinear-regression-1.html.

Northon, Karen. 2018. “Mars Helicopter to Fly on NASA’s Next Red Planet Rover Mission | NASA.” https://www.nasa.gov/press-release/mars-helicopter-to-fly-on-nasa-s-next-red-planet-rover-mission/.

Okamoto, M., K. Yasuda, and A. Azuma. 1996. “Aerodynamic Characteristics of the Wings and Body of a

Dragonfly." *Journal of Experimental Biology* 199 (2): 281–94.

Perminov, V. G. 1999. *The Difficult Road to Mars: A Brief History of Mars Exploration in the Soviet Union*. NASA History Division Office of Policy and Plans and Office of Space Science. https://ntrs.nasa.gov/search.jsp?R=19990054835.

Portree, David S. F. 2001. *Humans to Mars: Fifty Years of Mission Planning, 1950-2000*. https://ntrs.nasa.gov/search.jsp?R=20010020400.

Selig, Michael S., and James J. Guglielmo. 1997. "High-Lift Low Reynolds Number Airfoil Design." *Journal of Aircraft* 34 (1): 72–79. https://doi.org/10.2514/2.2137.

Shi, Yayun, Raphael Gross, Charles A. Mader, and Joaquim Martins. 2018. "Transition Prediction in a RANS Solver Based on Linear Stability Theory for Complex Three-Dimensional Configurations." In *2018 AIAA Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2018-0819.

Shrestha, Robin, Moble Benedict, Vikram Hrishikeshavan, and Inderjit Chopra. 2016. "Hover Performance of a Small-Scale Helicopter Rotor for Flying on Mars." *Journal of Aircraft* 53 (4): 1160–67. https://doi.org/10.2514/1.C033621.

Shyy, Wei, Yongsheng Lian, Jian Tang, Dragos Viieru, and Hao Liu. 2007. *Aerodynamics of Low Reynolds Number Flyers*. Cambridge University Press.

Srinivas, N., and Kalyanmoy Deb. 1994. "Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms." *Evolutionary Computation* 2 (3): 221–48. https://doi.org/10.1162/evco.1994.2.3.221.

Sullivan, J.M. 2006. "Evolution or Revolution? The Rise of UAVs." *IEEE Technology and Society Magazine* 25 (3): 43–49. https://doi.org/10.1109/MTAS.2006.1700021.

Sunada, S., T. Yasuda, K. Yasuda, and K. Kawachi. 2002. "Comparison of Wing Characteristics at an Ultralow Reynolds Number." *Journal of Aircraft* 39 (2): 331–38. https://doi.org/10.2514/2.2931.

Tangler, J L, and Cyrus Ostowari. 1991. "Horizontal Axis Wind Turbine Post Stall Airfoil Characteristics Synthesization," June, 7.

Tangler, J. L., and M. S. Selig. 1997. "An Evaluation of an Empirical Model for Stall Delay Due to Rotation for HAWTS." NREL/CP-440-23258; CONF-970608-2. National Renewable Energy Lab., Golden, CO (United States). https://www.osti.gov/biblio/513540.

Viterna, Larry A., and Robert D. Corrigan. 1982. "Fixed Pitch Rotor Performance of Large Horizontal Axis Wind Turbines." In . https://ntrs.nasa.gov/search.jsp?R=19830010962.

Vries, O. de. 1979. "Fluid Dynamic Aspects of Wind Energy Conversion." AGARD-AG-243. ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT NEUILLY-SUR-SEINE (FRANCE). https://apps.dtic.mil/docs/citations/ADA076315.

Weick, Fred Ernest. 1930. *Aircraft Propeller Design*. McGraw-Hill Book Company, Incorporated.

White, Frank M. 2016. *Fluid Mechanics*. Eighth edition. New York, NY: McGraw-Hill Education.

Yamauchi, G. K. Johnson. 1983. “Trends of Reynolds Number Effects on Two-Dimensional Airfoil Characteristics for Helicopter Rotor Analyses.” https://ntrs.nasa.gov/search.jsp?R=19830016201.

APPENDIX A: Additional Data Tables

Table A.1 JPL Reference Mars Atmosphere for -20° Latitude (Colozza et al. 2005)

Altitude (ft)	T(R)	P (lbf/ft^2)	Rho (Slug/ft^3)	Mu (slug/ft*s)
5.249344	439.2	13.637905	0.000027105991	0.0000002568855
21.653544	430.2	13.6316395	0.0000276628571	0.0000002527085
49.2126	428.4	13.6232855	0.0000277598721	0.00000025062
90.2231	426.6	13.608666	0.0000278491259	0.00000025062
147.6378	424.8	13.5898695	0.0000279267379	0.0000002485315
221.4567	423	13.5648075	0.0000279927081	0.0000002485315
311.6798	421.2	13.53348	0.0000280489768	0.000000246443
426.5092	421.2	13.4937985	0.0000279674842	0.000000246443
574.147	419.4	13.4436745	0.0000279830066	0.000000246443
779.1995	417.6	13.374754	0.0000279558424	0.0000002443545
1066.273	415.8	13.2765945	0.0000278724095	0.0000002443545
1476.378	414	13.136665	0.0000276997228	0.000000242266
2091.5355	410.4	12.9299035	0.0000275018122	0.0000002401775
2870.735	406.8	12.668841	0.0000271874836	0.000000238089
3690.945	408.6	12.401513	0.0000264947965	0.0000002401775
4511.155	410.4	12.138362	0.0000258215124	0.0000002401775
5331.365	412.2	11.883565	0.0000251676313	0.000000242266
6151.575	412.2	11.6350335	0.0000246398697	0.000000242266
6971.785	412.2	11.390679	0.0000241237499	0.000000242266
7791.995	412.2	11.15259	0.0000236173316	0.000000242266
8612.205	412.2	10.918678	0.0000231225551	0.000000242266
9432.415	412.2	10.688943	0.0000227344951	0.0000002401775
10252.625	410.4	10.463385	0.000022255241	0.0000002401775
11072.835	408.6	10.242004	0.0000218827034	0.0000002401775
11893.045	408.6	10.0248	0.0000214189717	0.0000002401775
12713.255	406.8	9.8138615	0.0000210580759	0.000000238089
13533.465	405	9.6050115	0.0000207010607	0.000000238089
14353.675	403.2	9.39825	0.0000203479261	0.000000238089
15173.885	403.2	9.197754	0.0000199113586	0.000000238089
15994.095	401.4	8.9993465	0.0000195718061	0.0000002360005
16814.305	399.6	8.805116	0.0000192361342	0.0000002360005

Table A.2 Experimental and XFOIL Polar data for E61 (Re = 40,000, t/c = 0.0567, camber = 6.69%) (Miley 1982)

Alpha (Deg)	Cl Exp	Cl XFOIL	Cd Exp	Cd XFOIL
0	0.47	0.1019	0.0438	0.05039
1	0.53	0.4516	0.0466	0.04254
2	0.6	0.5797	0.05	0.04199
3	0.66	0.6488	0.0536	0.04798
4	0.72	0.7077	0.0582	0.05543
5	0.78	0.8685	0.064	0.06245
6	0.85	1.0161	0.0715	0.06701
7	0.94	1.1521	0.0811	0.05823
8	1.06	1.2881	0.0683	0.04945
9	1.16	1.3506	0.0489	0.06006
10	1.25	1.4604	0.0372	0.07915
11	1.33	1.4037	0.0345	0.09765
12	1.35	1.3261	0.0407	0.11977
13	1.29	1.2642	---	---
14	1.21	1.2107	---	---
15	1.14	1.2002	---	---
16	1.11	1.2255	---	---
17	1.11	0.29463	---	---

Table A.3 Experimental and XFOIL Polar data for E61 (Re = 60,000, t/c = 0.0567, camber = 6.69%) (Miley 1982)

Alpha (Deg)	Cl Exp	Cl XFOIL	Cd Exp	Cd XFOIL
0	0.48	0.2688	0.0401	0.04138
1	0.56	0.4518	0.0407	0.04087
2	0.63	0.6348	0.0411	0.04036
3	0.71	0.7755	0.041	0.04414
4	0.78	0.9127	0.0395	0.04692
5	0.88	1.0525	0.0367	0.04812
6	0.99	1.3244	0.0341	0.02568
7	1.1	1.3205	0.0316	0.03661
8	1.21	1.3545	0.0291	0.04525
9	1.3	1.4931	0.0267	0.06099
10	1.38	1.5065	0.0265	0.07681
11	1.41	1.4319	0.0296	0.09341
12	1.38	1.3386	0.0363	0.11539
13	1.32	1.2553	---	---
14	1.24	1.2244	---	---
15	1.16	1.2112	---	---
16	1.13	1.198	---	---
17	1.12	1.1997	---	---
18	1.12	1.167	---	---

Table A.4 Experimental and XFOIL Polar data for NACA0009 (Re = 42,000, t/c = 0.09, camber = 0%) (Miley 1982)

Alpha (Deg)	Cl Exp	Cl XFOIL	Cd Exp	Cd XFOIL
0	-0.06	0	0.0179	0.01762
1	0.05	0.0504	0.0169	0.01802
2	0.17	0.1063	0.0169	0.01945
3	0.27	0.1838	0.0192	0.02288
4	0.36	0.5022	0.0224	0.02175
5	0.44	0.5646	0.0251	0.02614
6	0.51	0.6562	0.0272	0.03465
7	0.57	0.7277	0.029	0.04772
8	0.62	0.7543	---	---
9	0.67	0.6043	---	---
10	0.71	0.6481	---	---
11	0.74	0.6428	---	---
12	0.77	0.6538	---	---
13	0.77	0.7176	---	---
14	0.77	0.7219	---	---
15	0.75	0.7393	---	---
16	0.74	0.7644	---	---
17	0.74	0.7824	---	---
18	0.75	0.7966	---	---

Table A.5 Experimental and XFOIL Polar data for NACA0009 (Re = 60,000, t/c = 0.09, camber = 0%) (Miley 1982)

Alpha (Deg)	Cl Exp	Cl XFOIL	Cd Exp	Cd XFOIL
-5	-0.51	-0.5582	0.0181	0.02218
-4	-0.42	-0.49	0.0174	0.01729
-3	-0.31	-0.3907	0.0173	0.01778
-2	-0.21	-0.0999	0.0174	0.01714
-1	-0.11	-0.0445	0.0173	0.01549
0	0	0	0.017	0.01503
1	0.1	0.0446	0.0171	0.01549
2	0.21	0.1	0.0181	0.01714
3	0.31	0.3907	0.0201	0.01778
4	0.4	0.49	0.0233	0.01729
5	0.49	0.5582	0.0279	0.02218
6	0.57	0.6527	0.0343	0.02992
7	0.64	0.7344	0.0429	0.04069
8	0.68	0.7803	0.057	0.05658
9	0.72	0.6548	0.0761	0.1058
10	0.68	0.6422	0.1315	0.12054
11	0.64	0.6868	---	---
12	0.64	0.6794	---	---
13	0.64	0.6943	---	---
14	0.65	0.7204	---	---
15	0.65	0.7308	---	---
16	0.66	0.7672	---	---
17	0.67	0.7646	---	---

Table A.6 Experimental and XFOIL Polar data for GOE795 (Re = 17,000, t/c = 0.08, camber = 2.4%) (Miley 1982)

Alpha (Deg)	Cl Exp	Cl XFOIL	Cd Exp	Cd XFOIL
-3	-0.25	-0.2325	0.0544	0.03632
-2	-0.16	-0.1041	0.0512	0.02759
-1	-0.06	-0.0254	0.0475	0.028
0	0.04	0.0502	0.0438	0.02898
1	0.12	0.124	0.0408	0.03056
2	0.17	0.1952	0.0387	0.0328
3	0.22	0.2623	0.0405	0.03589
4	0.37	0.3317	0.0522	0.0404
5	0.61	0.5135	0.0668	0.0482
6	0.78	0.6719	0.0794	0.0545
7	0.82	0.8287	0.0878	0.05347
8	0.81	0.9274	0.0968	0.05485
9	0.79	0.9527	0.1072	0.0709
10	0.77	0.9176	0.1199	0.09093
11	0.75	0.8592	0.1379	0.11741
12	0.71	0.8347	0.1643	0.1404
13	0.66	0.8311	---	---
14	0.62	0.8342	---	---

Table A.7 Experimental and XFOIL Polar data for GOE795 (Re = 40,000, t/c = 0.08, camber = 2.4%) (Miley 1982)

Alpha (Deg)	Cl Exp	Cl XFOIL	Cd Exp	Cd XFOIL
-3	-0.11	-0.2601	0.0167	0.02984
-2	-0.02	-0.1525	0.0152	0.02282
-1	0.07	-0.0341	0.0164	0.01963
0	0.16	0.0414	0.0186	0.0209
1	0.26	0.116	0.0203	0.02273
2	0.39	0.1882	0.022	0.02524
3	0.54	0.255	0.0243	0.02871
4	0.69	0.4485	0.0273	0.03506
5	0.78	0.6404	0.0309	0.03846
6	0.85	0.8675	0.0352	0.02851
7	0.9	0.9416	0.0415	0.03356
8	0.95	1.0161	0.0534	0.04478
9	0.98	1.0466	0.0677	0.06051
10	0.97	1.0153	---	---
11	0.93	0.8894	---	---
12	0.87	0.7963	---	---
13	0.85	0.7874	---	---
14	0.85	0.8141	---	---
15	0.85	0.8277	---	---

Table A.8 Experimental and XFOIL Polar data for GOE795 (Re = 60,000, t/c = 0.08, camber = 2.4%) (Miley 1982)

Alpha (Deg)	Cl Exp	Cl XFOIL	Cd Exp	Cd XFOIL
-3	-0.1	-0.2544	0.0144	0.02544
-2	-0.01	-0.155	0.0143	0.02025
-1	0.09	-0.0402	0.016	0.01693
0	0.2	0.0354	0.0177	0.01834
1	0.33	0.1106	0.018	0.0203
2	0.47	0.2727	0.0179	0.02354
3	0.59	0.4488	0.0184	0.0259
4	0.69	0.6244	0.0192	0.02632
5	0.77	0.8046	0.0208	0.02126
6	0.84	0.8815	0.0218	0.02066
7	0.91	0.9356	0.0263	0.02805
8	0.99	1.0043	0.0476	0.03766
9	1.02	1.0531	0.0555	0.05357
10	1	1.0292	---	---
11	0.96	0.9671	---	---
12	0.91	0.8926	---	---
13	0.87	0.8181	---	---
14	0.86	0.8135	---	---
15	0.86	0.8293	---	---

Table A.9 Experimental and XFOIL Polar data for GOE796 (Re = 17,000, t/c = 0.12, camber = 3.68%) (Miley 1982)

Alpha (Deg)	Cl Exp	Cl XFOIL	Cd Exp	Cd XFOIL
-4	-0.26	-0.3858	0.0292	0.05939
-3	-0.19	-0.3075	0.0297	0.0508
-2	-0.09	-0.2114	0.0328	0.04452
-1	0	-0.1138	0.037	0.04046
0	0.06	-0.0324	0.0417	0.03974
1	0.12	0.0416	0.0471	0.04276
2	0.17	0.138	0.0534	0.04717
3	0.22	0.2811	0.0609	0.05377
4	0.27	0.3815	0.0698	0.05929
5	0.32	0.5041	0.0809	0.0659
6	0.37	0.6005	0.0948	0.07308
7	0.42	0.6345	0.1109	0.08274
8	0.47	0.7234	0.1282	0.09113
9	0.53	0.7515	0.1454	0.09801
10	0.58	0.7979	0.1595	0.11026
11	0.63	0.7931	0.1699	0.12456
12	0.66	0.8453	0.1781	0.1378

Table A.10 Experimental and XFOIL Polar data for GOE797 (Re = 17,000, t/c = 0.16, camber = 4.9%) (Miley 1982)

Alpha (Deg)	Cl Exp	Cl XFOIL	Cd Exp	Cd XFOIL
-4	-0.22	-0.453	0.0425	0.08739
-3	-0.14	-0.4027	0.0513	0.07582
-2	-0.07	-0.3372	0.0598	0.07301
-1	-0.02	-0.2368	0.0669	0.06424
0	0.04	-0.1122	0.0727	0.06271
1	0.1	0.02	0.0779	0.065175
2	0.15	0.1522	0.0832	0.06764
3	0.2	0.2423	0.089	0.07344
4	0.25	0.3435	0.0952	0.07888
5	0.31	0.4818	0.1016	0.08745
6	0.35	0.4791	0.1079	0.09425
7	0.38	0.5298	0.1141	0.1033
8	0.4	0.5887	0.1198	0.11464
9	0.4	0.6472	0.1253	0.12606
10	0.4	0.6434	0.1306	0.13276
11	0.4	0.7149	0.136	0.14141
12	0.4	0.7476	0.1417	0.14592
13	0.4	0.8267	0.1476	0.15649
14	0.39	0.8058	---	---
15	0.39	0.825	---	---
16	0.39	0.8245	---	---

Table A.11 Experimental and XFOIL Polar data for GOE801 (Re = 21,000, t/c = 0.0979, camber = 6.17%) (Miley 1982)

Alpha (Deg)	Cl Exp	Cl XFOIL	Cd Exp	Cd XFOIL
-3	-0.15	-0.3017	0.061	0.08004
-2	-0.06	-0.1061	0.0555	0.07676
-1	0.04	0.0813	0.0535	0.06527
0	0.13	0.2404	0.0543	0.06027
1	0.23	0.3895	0.0572	0.05978
2	0.31	0.5066	0.0617	0.06211
3	0.38	0.6337	0.0674	0.06186
4	0.43	0.7	0.0742	0.06715
5	0.47	0.7511	0.0822	0.0721
6	0.5	0.8645	0.0919	0.08026
7	0.52	0.8301	0.1035	0.093
8	0.54	0.8819	0.1164	0.10159
9	0.56	0.9194	0.1285	0.11043
10	0.58	0.9432	0.1399	0.12191
11	0.6	0.9982	0.1517	0.1359
12	0.61	0.9596	0.1646	0.14979
13	0.63	0.974	---	---
14	0.65	1.033	---	---
15	0.67	1.0169	---	---
16	0.69	1.017	---	---
17	0.71	1.0414	---	---

APPENDIX B: Sample Code

Code listed here is all the final code used in this project. All is done using MATLAB R2017b or later.

```
%Main BEMT Function Code

function [T, P, Q, Eff, MachMax, Comp, ReMax] =
Prop(Ni,RPM,Nb,R,Rhub,h,V,AlphaIdeal,E,CF)

    %Set up calculations

    ID = 1:1:Ni;                        %Blade Section Vector
    ID = ID';                           %Transposes it for easier use
    Temp = 422 - 0.00131*h;             %Temp from the 2005 Paper (R)
    Pr = 13.6 - 0.000294*h;             %Pressure for based on 2005 Paper
(lbf/ft^2)
    Rho = 0.0000282-5.52e-10*h;         %Formula for Rho based on SSA Report
(slug/ft^3)
    Mu = 2.47e-7 - 6.98e-13*h;          %Mu from 2005 Paper (slug/ft*s)
    %Temp = 520;                        %Temp for thinfoil (R)
    %Pr = 14;                           %Pressure for thinfoil(lbf/ft^2)
    %Rho = 0.00003240334957;            %Rho for thinfoil
    %Mu = 3.7903436e-7;                 %Mu for thinfoil
    MuVis = Mu/Rho;                     %Air Kinetic Viscosity (units)
    SoS = 38.33298*sqrt(Temp);          %Speed of Sound (ft/s)
    th = 0.05;                          %Thickness

    w = zeros(Ni,1);                    %Initial Induced Veclocity (ft/s)
    w = w+1;                            %Gives w a value of 1
    DeltaR = (R-Rhub)/Ni;               %Hub Widths (ft)
    n = RPM/60;                         %Revolutions in (rev/s)
    Omega = RPM*pi/30;                  %Radial Speed (rad/s)
    r = zeros(Ni,1);                    %Sets our initial radial position vectors

    %Set up for our radial position vector

    r(1) = Rhub + (DeltaR/2);           %Sets the first radial position
    m = 2;                              %Sets first variable for the loop
    while m <= Ni
        r(m) = r(m-1)+DeltaR;           %Sets individual values
        m = m + 1;                      %Moves to next value
    end

    %Calculations of Key Values

    x = r/R;                                        %Radial Station as a
Percentage
    %c = 0.1666667*(x./x);                           %Chord function for
thinfoil (ft)
```

```
    c = CF*R*(-0.101+(2.27.*x)+(-7.14.*(x.^2))+(8.38.*(x.^3))+(-3.37.*(x.^4)));
%Chord Function for S1223Mod3 Airfoil
    ctr = c./r;                                         %Chord to radial position
ratio
    K = (0.1517./(ctr)).^(1/1.084);                     %K value for Cl correction
    DeltaA = DeltaR*c;                                  %Area of Each Radial
Station (ft^2)
    AR = (R^2)/sum(DeltaA);
    Omegar = Omega*r;                                   %Gives Radial Station
Linear Velocities (ft/s)
    Vr = sqrt((Omegar.^2)+(V^2));                       %Air Relative Speed (ft/s)
    M = Vr/SoS;                                         %Mach Number
    PhiRad = atan(V./Omegar);                           %Phi (rad)
    Phi = PhiRad*180/pi;                                %Phi (deg)
    Beta = AlphaIdeal + Phi;                            %Beta (deg)
    %Beta = AlphaIdeal;
    BetaRad = Beta*pi/180;                              %Beta (rad)
    Alphai = zeros(Ni,1);                               %Alphai (deg)
    AlphaiRad = Alphai*pi/180;                          %Alphai (rad)
    wdif = 1;                                           %Initial wdif (ft/s)

    %Induced Velocity Calculation Loop
    while wdif > 0.0001
        Ve = sqrt((Omegar.^2)+((w+V).^2));                          %Give
Effective Velocity (ft/s)
        Re = c.*Ve.*(Rho/Mu);                                       %Reynolds
Number
        %[Alphamax, Alphazi, AlphaSlope] = AlphaInfoSimFoil (Re, Ni);  %Alpha
Curve Data
        [Alphamax, Alphazi, AlphaSlope] = AlphaInfoS1223Mod3 (Re, Ni);
        AlphamaxRad = Alphamax*pi/180;                              %Converts
Alphamax to radians (rad)
        AlphaziRad = Alphazi*pi/180;                                %Converts
Alphazi to radians (rad)
        DeltaAlpha = (Alphamax - Alphazi).*((((K.*ctr)./0.136).^1.6)-1);
%Alpha Rotation Correction
        Alpha = Beta - Phi - Alphai + Alphazi + DeltaAlpha;         %Alpha
(deg)
        AlphaRad = Alpha.*pi./180;                                  %Alpha
(rad)
        %[Cl, Cd, Cdp, AlphaStall] = CoeffSimFoil (Re, Alpha, AlphaRad, Alphazi,
Alphamax,DeltaAlpha, AlphaSlope, AR, Ni, AlphaIdeal);
        [Cl, Cd] = CoeffS1223Mod3 (Re, Alpha, AlphaRad, Alphazi,
Alphamax,DeltaAlpha, AlphaSlope, AR, Ni);
        fw = ((w.*8*pi.*r)./(Nb.*c))-((Ve./(V+w)).*((Cl.*Omegar)-(Cd.*(w+V))));
%f(w)
        dfw =
((8*pi*r)./(Nb*c))-(Cl.*Omegar.*((1./Ve)-(Ve./((V+w).^2))))+(Cd.*((V+w)./Ve));
%f'(w)
```

```
        wnew = w - (fw./dfw);                                   %New
induced velocity value (ft/s)
        wdif = wnew-w;                                          %Difference
in induce velocity values
        AlphaiRad = atan(wnew./Ve);                             %Alphai
(rad)
        Alphai = AlphaiRad*180/pi;                              %Alphai
(deg)
        w = wnew;                                               %Sets new
induced velocity
    end


    %Tip/Hub Corrections
    TSR = (Omega*R)./(mean(w)+V);                       %Estimated Tip Speed
Ratio
    g = exp(-0.125*((Nb*TSR)-21));                      %g Correction factor
    Ptip = g*(Nb/2)*((R-r)./(r.*sin(PhiRad)));
    Ftip = (2/pi)*acos(exp(-1*Ptip));
    Phub = g*(Nb/2)*((r-Rhub)./(r.*sin(PhiRad)));
    Fhub = (2/pi)*acos(exp(-1*Phub));
    Fp = Ftip .* Fhub;

    %Sectional Values
    M = Ve/SoS;                                 %Mach Number Calculation
    CombinedAng = PhiRad+AlphaiRad;             %Combined angle for Thrust, Power,
and Torque calcs (rad)
    Re = c.*Ve.*(Rho/Mu);                       %Reynolds Number
    ReMax = Re(length(Re));                     %Maximum Reynolds Number

    %Compression functions here
    m = 1;
    while m <= Ni
        if M(m,1) < 0.70
            Cl(m) = Cl(m)/sqrt(1-(M(m)^2));         %Cl Compression Correction
            Comp = 0;                               %Says Compression Occured
        else
            Comp = 1;                               %Says Compression did not
occure
        end
        m = m+1;
    end

    dl = 0.5*DeltaA.*Rho.*Cl.*Ve.*Ve;                               %Lift of
Sectional Area (lbf)
    dd = 0.5*DeltaA.*Rho.*Cd.*Ve.*Ve;                               %Drag of
Sectional Area (lbf)
    dT = Fp.*((dl.*cos(CombinedAng))-(dd.*sin(CombinedAng)));       %Thrust of
Sectional Area (lbf)
```

```
    dQ = Fp.*r.*((dl.*sin(CombinedAng))+(dd.*cos(CombinedAng)));      %Torque of
Sectional Area (lbf*ft)
    dP = Omega.*dQ;                                                   %Power of
Sectional Area (lbf*ft/s)

    %Totaled Important Values

    T = Nb*sum(dT);                             %Total Thrust (lbf)
    Q = Nb*sum(dQ);                             %Total Torque (lbf)
    P = Nb*sum(dP)/550;                         %Total Power (BHP)
    Ct = T/(Rho*(n^2)*((2*R)^4));               %Thrust Coefficient
    Cq = Q/(Rho*(n^2)*((2*R)^5));               %Torque Coefficient
    Cp = (P*550)/(Rho*(n^3)*((2*R)^5));         %Power Coefficient
    J = V/(n*2*R);                              %Advance Ratio
    Nup = J*(Ct/Cp);                            %Propeller Efficiency
    Eff = T./P;                                 %Thrust to Power Ratio
    MachMax = M(length(M));                     %Gives the Maximum Mach Value

    %Bending Moment Calculation (Unused)
    %Width = 0.3048*mean(c);
    %Height = 0.03*Width;
    %Length = 0.3048*(R-Rhub);
    %Metricr = 0.3048*r;

    %I = Width*(Height^3)/12;
    %Delt = (((4.44*dT).*(Metricr.^2))./(6*E*I)).*((3*Length)-Metricr);
    %DeltaMax = sum(Delt)/0.3048;


end


%Contains Angle of Attack Stats for S1223Mod3 Airfoil

function [Alphamax, Alphazi, AlphaSlope] = AlphaInfoS1223Mod3 (Re, Ni)

    %Sets the initial values for the Alpha info
    Alphamax = zeros(Ni,1);          %Sets the Alphamax Vector
    Alphazi = zeros(Ni,1);           %Sets the Alphazi Vector
    AlphaSlope = zeros(Ni,1);        %Sets the AlphaSlope Vector
    Alphamax1000 = 24.5196;          %Sets the Alphamax for Re = 1000
    Alphazi1000 = -0.1713;           %Sets the Alphazi for Re = 1000
    AlphaSlope1000 = 0.047328;       %Sets the AlphaSlope for Re = 1000
    Alphamax4000 = 25.9310;          %Sets the Alphamax for Re = 4000
    Alphazi4000 = 1.0317;            %Sets the AlphaSlope for Re = 4000
    AlphaSlope4000 = 0.034196;       %Sets the AlphaSlope for Re = 4000
    Alphamax7000 = 21.9364;          %Sets the Alphamax for Re = 7000
    Alphazi7000 = 0.8174;            %Sets the AlphaSlope for Re = 7000
    AlphaSlope7000 = 0.030476;       %Sets the AlphaSlope for Re = 7000
    Alphamax10000 = 18.3349;         %Sets the Alphamax for Re = 10000
```

```
    Alphazi10000 = 0.3844;          %Sets the Alphazi for Re = 10000
    AlphaSlope10000 = 0.033233;     %Sets the AlphaSlope for Re = 10000
    Alphamax13000 = 18.3003;        %Sets the Alphamax for Re = 13000
    Alphazi13000 = 0.3402;          %Sets the Alphazi for Re = 13000
    AlphaSlope13000 = 0.037133;     %Sets the AlphaSlope for Re = 13000
    Alphamax16000 = 19.7343;        %Sets the Alphamax for Re = 16000
    Alphazi16000 = 0.3012;          %Sets the Alphazi for Re = 16000
    AlphaSlope16000 = 0.038388;     %Sets the AlphaSlope for Re = 16000

    %Loop to interpolate values
    n = 1;                          %Sets first variable for the loop
    while n <= Ni
        if Re(n) <= 1000
            Alphamax(n,1) = Interpolate(0,Re(n),1000,0,Alphamax1000);
            Alphazi(n,1) = Interpolate(0,Re(n),1000,0,Alphazi1000);
            AlphaSlope(n,1) = Interpolate(0,Re(n),1000,0,AlphaSlope1000);
        elseif Re(n) > 1000 && Re(n) <= 4000
            Alphamax(n,1) =
Interpolate(1000,Re(n),4000,Alphamax1000,Alphamax4000);
            Alphazi(n,1) = Interpolate(1000,Re(n),4000,Alphazi1000,Alphazi4000);
            AlphaSlope(n,1) =
Interpolate(1000,Re(n),4000,AlphaSlope1000,AlphaSlope4000);
        elseif Re(n) > 4000 && Re(n) <= 7000
            Alphamax(n,1) =
Interpolate(4000,Re(n),7000,Alphamax4000,Alphamax7000);
            Alphazi(n,1) = Interpolate(4000,Re(n),7000,Alphazi4000,Alphazi7000);
            AlphaSlope(n,1) =
Interpolate(4000,Re(n),7000,AlphaSlope4000,AlphaSlope7000);
        elseif Re(n) > 7000 && Re (n) <= 10000
            Alphamax(n,1) =
Interpolate(7000,Re(n),10000,Alphamax7000,Alphamax10000);
            Alphazi(n,1) =
Interpolate(7000,Re(n),10000,Alphazi7000,Alphazi10000);
            AlphaSlope(n,1) =
Interpolate(7000,Re(n),10000,AlphaSlope7000,AlphaSlope10000);
        elseif Re(n) > 10000 && Re (n) <= 13000
            Alphamax(n,1) =
Interpolate(10000,Re(n),13000,Alphamax10000,Alphamax13000);
            Alphazi(n,1) =
Interpolate(10000,Re(n),13000,Alphazi10000,Alphazi13000);
            AlphaSlope(n,1) =
Interpolate(10000,Re(n),13000,AlphaSlope10000,AlphaSlope13000);
        elseif Re(n) > 13000
            Alphamax(n,1) =
Interpolate(13000,Re(n),16000,Alphamax13000,Alphamax16000);
            Alphazi(n,1) =
Interpolate(13000,Re(n),16000,Alphazi13000,Alphazi16000);
            AlphaSlope(n,1) =
Interpolate(13000,Re(n),16000,AlphaSlope13000,AlphaSlope16000);
        end
```

```
        n = n+1;
    end

%Function for calculating Cl and Cd coefficients for S1223Mod3 Airfoil

function [Cl, Cd] = CoeffS1223Mod3 (Re, Alpha, AlphaRad, Alphazi,
Alphamax,DeltaAlpha, AlphaSlope, AR, Ni)
    AlphaStall = Alphamax + DeltaAlpha;
    AlphaStallRad = AlphaStall*(pi/180);
    Cl = zeros(Ni,1);                %Sets up Cl Vector
    Cd = zeros(Ni,1);                %Sets up Cd Vector
    Cls = zeros(Ni,1);               %Sets up Cls Vector
    Cds = zeros(Ni,1);               %Sets up Cds Vector

    %Set values for Re = 1000
    Cl1000 =
(0.133806358576094)+(0.0823945242036631.*Alpha)+(0.00157270163958108.*(Alpha.^2))
+(-0.000359089570225581.*(Alpha.^3))+(1.42088492881524e-05.*(Alpha.^4))+(-2.26341
318523283e-07.*(Alpha.^5))+(1.10776021223705e-09.*(Alpha.^6));
    Cd1000 =
(0.115978480753278)+(-0.000604212204483838.*Alpha)+(0.000538600252756660.*(Alpha.
^2))+(5.31464769881630e-06.*(Alpha.^3))+(5.33239379722895e-07.*(Alpha.^4))+(-4.00
381651178457e-08.*(Alpha.^5))+(4.81997287626680e-10.*(Alpha.^6));
    Cls1000 =
(0.133806358576094)+(0.0823945242036631.*AlphaStall)+(0.00157270163958108.*(Alpha
Stall.^2))+(-0.000359089570225581.*(AlphaStall.^3))+(1.42088492881524e-05.*(Alpha
Stall.^4))+(-2.26341318523283e-07.*(AlphaStall.^5))+(1.10776021223705e-09.*(Alpha
Stall.^6));
    Cds1000 =
(0.115978480753278)+(-0.000604212204483838.*AlphaStall)+(0.000538600252756660.*(A
lphaStall.^2))+(5.31464769881630e-06.*(AlphaStall.^3))+(5.33239379722895e-07.*(Al
phaStall.^4))+(-4.00381651178457e-08.*(AlphaStall.^5))+(4.81997287626680e-10.*(Al
phaStall.^6));

     %Set values for Re = 4000
```

```
    Cl4000 =
(0.166518496557882)+(0.0890638136358078.*Alpha)+(0.00546962380669993.*(Alpha.^2))
+(-0.000567259234106545.*(Alpha.^3))+(-2.25625232208894e-05.*(Alpha.^4))+(2.72717
077012937e-06.*(Alpha.^5))+(-5.49743310249514e-08.*(Alpha.^6));
    Cd4000 =
(0.0661653145119418)+(0.000626667181515809.*Alpha)+(0.00115401316779416.*(Alpha.^
2))+(-2.24145467805783e-05.*(Alpha.^3))+(-3.68040360208659e-06.*(Alpha.^4))+(3.11
798822033536e-07.*(Alpha.^5))+(-6.45620165735217e-09.*(Alpha.^6));
    Cls4000 =
(0.166518496557882)+(0.0890638136358078.*AlphaStall)+(0.00546962380669993.*(Alpha
Stall.^2))+(-0.000567259234106545.*(AlphaStall.^3))+(-2.25625232208894e-05.*(Alph
aStall.^4))+(2.72717077012937e-06.*(AlphaStall.^5))+(-5.49743310249514e-08.*(Alph
aStall.^6));
    Cds4000 =
(0.0661653145119418)+(0.000626667181515809.*AlphaStall)+(0.00115401316779416.*(Al
phaStall.^2))+(-2.24145467805783e-05.*(AlphaStall.^3))+(-3.68040360208659e-06.*(A
lphaStall.^4))+(3.11798822033536e-07.*(AlphaStall.^5))+(-6.45620165735217e-09.*(A
lphaStall.^6));

    %Set values for Re = 7000
    Cl7000 =
(-0.0808093423690828)+(0.141396475583597.*Alpha)+(0.0291904368497246.*(Alpha.^2))
+(-0.00710827191628144.*(Alpha.^3))+(0.000551024960074883.*(Alpha.^4))+(-1.834560
21635572e-05.*(Alpha.^5))+(2.22600427088553e-07.*(Alpha.^6));
    Cd7000 =
(0.0599955900482245)+(-0.00233932572857232.*Alpha)+(0.00341308410863779.*(Alpha.^
2))+(-0.000538725128181817.*(Alpha.^3))+(4.56733966492944e-05.*(Alpha.^4))+(-1.69
952597552471e-06.*(Alpha.^5))+(2.22390739629217e-08.*(Alpha.^6));
    Cls7000 =
(-0.0808093423690828)+(0.141396475583597.*AlphaStall)+(0.0291904368497246.*(Alpha
Stall.^2))+(-0.00710827191628144.*(AlphaStall.^3))+(0.000551024960074883.*(AlphaS
tall.^4))+(-1.83456021635572e-05.*(AlphaStall.^5))+(2.22600427088553e-07.*(AlphaS
tall.^6));
```

```
    Cds7000 =
(0.0599955900482245)+(-0.00233932572857232.*AlphaStall)+(0.00341308410863779.*(Al
phaStall.^2))+(-0.000538725128181817.*(AlphaStall.^3))+(4.56733966492944e-05.*(Al
phaStall.^4))+(-1.69952597552471e-06.*(AlphaStall.^5))+(2.22390739629217e-08.*(Al
phaStall.^6));

    %Set values for Re = 10000
    Cl10000 =
(0.100666959393033)+(0.121245300032986.*Alpha)+(0.0119468230103778.*(Alpha.^2))+(
-0.00167838950614716.*(Alpha.^3))+(-6.88683980191093e-05.*(Alpha.^4))+(1.26724569
841708e-05.*(Alpha.^5))+(-3.46275632241685e-07.*(Alpha.^6));               %Sets
up Cl Vector
    Cd10000 =
(0.0618798110314736)+(-0.000248264123428146.*Alpha)+(0.000813968635056768.*(Alpha
.^2))+(3.33719501170350e-05.*(Alpha.^3))+(-1.16346287400465e-06.*(Alpha.^4))+(-1.
46139205571484e-07.*(Alpha.^5))+(5.82334987476293e-09.*(Alpha.^6));
%Sets up Cd Vector
    Cls10000 =
(0.100666959393033)+(0.121245300032986.*AlphaStall)+(0.0119468230103778.*(AlphaSt
all.^2))+(-0.00167838950614716.*(AlphaStall.^3))+(-6.88683980191093e-05.*(AlphaSt
all.^4))+(1.26724569841708e-05.*(AlphaStall.^5))+(-3.46275632241685e-07.*(AlphaSt
all.^6));
    Cds10000 =
(0.0618798110314736)+(-0.000248264123428146.*AlphaStall)+(0.000813968635056768.*(
AlphaStall.^2))+(3.33719501170350e-05.*(AlphaStall.^3))+(-1.16346287400465e-06.*(
AlphaStall.^4))+(-1.46139205571484e-07.*(AlphaStall.^5))+(5.82334987476293e-09.*(
AlphaStall.^6));

    %Set values for Re = 13000
    Cl13000 =
(0.118110877495861)+(0.195762808101148.*Alpha)+(0.00949369395937878.*(Alpha.^2))+
(-0.00636614923046947.*(Alpha.^3))+(0.000682461982988315.*(Alpha.^4))+(-2.8898877
5192877e-05.*(Alpha.^5))+(4.31422248763872e-07.*(Alpha.^6));               %Sets
up Cl Vector
```

```
    Cd13000 =
(0.0585863348900040)+(-0.000566283701953687.*Alpha)+(0.000955242137686517.*(Alpha
.^2))+(6.48654899936202e-05.*(Alpha.^3))+(-1.08594253090165e-05.*(Alpha.^4))+(6.2
9653425336882e-07.*(Alpha.^5))+(-1.36189264897380e-08.*(Alpha.^6));
    Cls13000 =
(0.118110877495861)+(0.195762808101148.*AlphaStall)+(0.00949369395937878.*(AlphaS
tall.^2))+(-0.00636614923046947.*(AlphaStall.^3))+(0.000682461982988315.*(AlphaSt
all.^4))+(-2.88988775192877e-05.*(AlphaStall.^5))+(4.31422248763872e-07.*(AlphaSt
all.^6));
    Cds13000 =
(0.0585863348900040)+(-0.000566283701953687.*AlphaStall)+(0.000955242137686517.*(
AlphaStall.^2))+(6.48654899936202e-05.*(AlphaStall.^3))+(-1.08594253090165e-05.*(
AlphaStall.^4))+(6.29653425336882e-07.*(AlphaStall.^5))+(-1.36189264897380e-08.*(
AlphaStall.^6));

    %Set values for Re = 16000
    Cl16000 =
(0.156524698481090)+(0.174265748107204.*Alpha)+(0.00902767533827675.*(Alpha.^2))+
(-0.00510025556824186.*(Alpha.^3))+(0.000505787962051951.*(Alpha.^4))+(-1.9761423
7188371e-05.*(Alpha.^5))+(2.67711742253585e-07.*(Alpha.^6));               %Sets
up Cl Vector
    Cd16000 =
(0.0534300212971274)+(-0.00207311968802088.*Alpha)+(0.00150787007863827.*(Alpha.^
2))+(0.000157623876306690.*(Alpha.^3))+(-3.75394158083362e-05.*(Alpha.^4))+(2.408
23546088152e-06.*(Alpha.^5))+(-5.00708703842964e-08.*(Alpha.^6));
%Sets up Cd Vector
    Cls16000 =
(0.156524698481090)+(0.174265748107204.*AlphaStall)+(0.00902767533827675.*(AlphaS
tall.^2))+(-0.00510025556824186.*(AlphaStall.^3))+(0.000505787962051951.*(AlphaSt
all.^4))+(-1.97614237188371e-05.*(AlphaStall.^5))+(2.67711742253585e-07.*(AlphaSt
all.^6));
    Cds16000 =
(0.0534300212971274)+(-0.00207311968802088.*AlphaStall)+(0.00150787007863827.*(Al
phaStall.^2))+(0.000157623876306690.*(AlphaStall.^3))+(-3.75394158083362e-05.*(Al
```

```
phaStall.^4))+(2.40823546088152e-06.*(AlphaStall.^5))+(-5.00708703842964e-08.*(Al
phaStall.^6));

    %Loop for interpolating the lift and drag values
    n = 1;                                  %Sets first variable for the loop
    while n <= Ni
        if Re(n) <= 1000
            Cl(n,1) = Interpolate(0,Re(n),1000,0,Cl1000(n));
            Cd(n,1) = Interpolate(0,Re(n),1000,0,Cd1000(n));
            Cls(n,1) = Interpolate(0,Re(n),1000,0,Cls1000(n));
            Cds(n,1) = Interpolate(0,Re(n),1000,0,Cds1000(n));
        elseif Re(n) > 1000 && Re(n) <= 4000
            Cl(n,1) = Interpolate(1000,Re(n),4000,Cl1000(n),Cl4000(n));
            Cd(n,1) = Interpolate(1000,Re(n),4000,Cd1000(n),Cd4000(n));
            Cls(n,1) = Interpolate(1000,Re(n),4000,Cls1000(n),Cls4000(n));
            Cds(n,1) = Interpolate(1000,Re(n),4000,Cds1000(n),Cds4000(n));
        elseif Re(n) > 4000 && Re(n) <= 7000
            Cl(n,1) = Interpolate(4000,Re(n),7000,Cl4000(n),Cl7000(n));
            Cd(n,1) = Interpolate(4000,Re(n),7000,Cd4000(n),Cd7000(n));
            Cls(n,1) = Interpolate(4000,Re(n),7000,Cls4000(n),Cls7000(n));
            Cds(n,1) = Interpolate(4000,Re(n),7000,Cds4000(n),Cds7000(n));
        elseif Re(n) > 7000 && Re(n) <= 10000
            Cl(n,1) = Interpolate(7000,Re(n),10000,Cl7000(n),Cl10000(n));
            Cd(n,1) = Interpolate(7000,Re(n),10000,Cd7000(n),Cd10000(n));
            Cls(n,1) = Interpolate(7000,Re(n),10000,Cls7000(n),Cls10000(n));
            Cds(n,1) = Interpolate(7000,Re(n),10000,Cds7000(n),Cds10000(n));
        elseif Re(n) > 10000 && Re(n) <= 13000
            Cl(n,1) = Interpolate(10000,Re(n),13000,Cl10000(n),Cl13000(n));
            Cd(n,1) = Interpolate(10000,Re(n),13000,Cd10000(n),Cd13000(n));
            Cls(n,1) = Interpolate(10000,Re(n),13000,Cls10000(n),Cls13000(n));
            Cds(n,1) = Interpolate(10000,Re(n),13000,Cds10000(n),Cds13000(n));
        elseif Re(n) > 13000
            Cl(n,1) = Interpolate(13000,Re(n),16000,Cl13000(n),Cl16000(n));
            Cd(n,1) = Interpolate(13000,Re(n),16000,Cd13000(n),Cd16000(n));
```

```
            Cls(n,1) = Interpolate(13000,Re(n),16000,Cls13000(n),Cls16000(n));
            Cds(n,1) = Interpolate(13000,Re(n),16000,Cds13000(n),Cds16000(n));
        end
        n = n+1;
    end

    %First calculate the custom correction, then the Corrigan
    %Schilling Correction
    b1 = 1.43656968592719;
    b2 = 1.75545378442278;
    b3 = 0.518367841387982;
    b4 = 2.14917276732139;
    b5 = -2.41326118040382;
    b6 = 1.85996056350822;
    b7 = 3.26463808395347;
    b8 = -2.96910717977840;
    b9 = 2.28183150338262;
    b10 = -1.10158216973537;
    b11 =  0.06048150937307982;
    b12 = 0.02;
    b13 =  0.07;

    A = (Alpha - Alphazi)./(AlphaStall-Alphazi);
    fAlpha = (sin((b1.*A) + b2).*sin((b3.*A) + b4).*sin((b5.*A) +
b6).*sin((b7.*A) + b8).*sin((b9.*A) + b10))+1;
    Cl = b11.*(Re.^b12).*(0.01.^b13).*fAlpha.*Cl;
    Cl = Cl + (DeltaAlpha.*AlphaSlope);
    fAlpha = (sin(b1+b2).*sin(b3+b4).*sin(b5+b6).*sin(b7+b8).*sin(b9+b10))+1;
    Cls = b11.*(Re.^b12).*(0.01.^b13).*fAlpha.*Cls;
    Cls = Cls + (DeltaAlpha.*AlphaSlope);


    %Here do the post-stall Viterna-Corrigan correction
    %First calculate A1, B1, etc. then loop for rest
```

```
    B1 =  (1+(0.065*AR))/0.91; %Alternatively 1.11+0.018*AR;
    A1 = B1/2;
    A2 = Cls-(B1.*sin(AlphaStallRad).*cos(AlphaStallRad));
    B2 = Cds-((B1.*(sin(AlphaStallRad).^2))./cos(AlphaStallRad));

    n = 1;
    while n <= Ni
      if Alpha(n)> AlphaStall(n)
          Cl(n,1) =
(A1.*sin(2.*AlphaRad(n)))+(A2(n,1).*((cos(AlphaRad(n)).^2)./sin(AlphaRad(n))));
          Cd(n,1) = (B1.* (sin(AlphaRad(n)).^2)) + (B2(n,1).*cos(AlphaRad(n)));
      end
      n = n + 1;
    end
end

%Contains Angle of Attack Stats for the Recreated Study Airfoil

function [Alphamax, Alphazi, AlphaSlope] = AlphaInfoSimFoil (Re, Ni)

    %Sets the initial values for the Alpha info
    Alphamax = zeros(Ni,1);          %Sets the Alphamax Vector
    Alphazi = zeros(Ni,1);           %Sets the Alphazi Vector
    AlphaSlope = zeros(Ni,1);        %Sets the AlphaSlope Vector
    Alphamax1000 = 11.5;             %Sets the Alphamax for Re = 1000
    Alphazi1000 = -0.5;              %Sets the Alphazi for Re = 1000
    AlphaSlope1000 = 0.05393;        %Sets the AlphaSlope for Re = 1000
    Alphamax3000 = 9.4;              %Sets the Alphamax for Re = 3000
    Alphazi3000 = 0.1;               %Sets the AlphaSlope for Re = 3000
    AlphaSlope3000 = 0.06397;        %Sets the AlphaSlope for Re = 3000
    Alphamax5000 = 8;                %Sets the Alphamax for Re = 5000
    Alphazi5000 = 0.2;               %Sets the AlphaSlope for Re = 5000
    AlphaSlope5000 = 0.0797;         %Sets the AlphaSlope for Re = 5000
```

```
    %Loop to interpolate values
    n = 1;                          %Sets first variable for the loop
    while n <= Ni
        if Re(n) <= 1000
            Alphamax(n,1) = Interpolate(0,Re(n),1000,0,Alphamax1000);
            Alphazi(n,1) = Interpolate(0,Re(n),1000,0,Alphazi1000);
            AlphaSlope(n,1) = Interpolate(0,Re(n),1000,0,AlphaSlope1000);
        elseif Re(n) > 1000 && Re(n) <= 3000
            Alphamax(n,1) =
Interpolate(1000,Re(n),3000,Alphamax1000,Alphamax3000);
            Alphazi(n,1) = Interpolate(1000,Re(n),3000,Alphazi1000,Alphazi3000);
            AlphaSlope(n,1) =
Interpolate(1000,Re(n),3000,AlphaSlope1000,AlphaSlope3000);
        elseif Re(n) > 3000
            Alphamax(n,1) =
Interpolate(3000,Re(n),5000,Alphamax3000,Alphamax5000);
            Alphazi(n,1) = Interpolate(3000,Re(n),5000,Alphazi3000,Alphazi5000);
            AlphaSlope(n,1) =
Interpolate(3000,Re(n),5000,AlphaSlope3000,AlphaSlope5000);
        end
        n = n+1;
    end

%Function for calculating Cl and Cd coefficients for Recreated Study Airfoil

function [Cl, Cd, Cdp, AlphaStall] = CoeffSimFoil (Re, Alpha, AlphaRad, Alphazi,
Alphamax,DeltaAlpha, AlphaSlope, AR, Ni, AlphaIdeal)
    AlphaStall = Alphamax + DeltaAlpha;
    AlphaStallRad = AlphaStall*(pi/180);
    Cl = zeros(Ni,1);               %Sets up Cl Vector
    Cd = zeros(Ni,1);               %Sets up Cd Vector
    Cdp = zeros(Ni,1);              %Sets up Cdp Vector
    Cls = zeros(Ni,1);              %Sets up Cls Vector
```

```
    Cds = zeros(Ni,1);              %Sets up Cds Vector

    %Set values for Re = 1000
    Cl1000 =
0.123+(0.169.*Alpha)+(-0.0127.*(Alpha.^2))+(0.000494.*(Alpha.^3))+(-0.00000687.*(
Alpha.^4));               %Sets up Cl Vector
    Cd1000 =
0.104+(-0.0024.*Alpha)+(0.000985.*(Alpha.^2))+(-0.0000139.*(Alpha.^3))+(-0.000000
26.*(Alpha.^4));               %Sets up Cd Vector
    Cdp1000 =
0.0511+(-0.0201.*Alpha)+(0.00425.*(Alpha.^2))+(-0.000206.*(Alpha.^3))+(0.00000343
.*(Alpha.^4));               %Sets up Cdp Vector
    Cls1000 =
0.123+(0.169.*AlphaStall)+(-0.0127.*(AlphaStall.^2))+(0.000494.*(AlphaStall.^3))+
(-0.00000687.*(AlphaStall.^4));
    Cds1000 =
0.104+(-0.0024.*AlphaStall)+(0.000985.*(AlphaStall.^2))+(-0.0000139.*(AlphaStall.
^3))+(-0.00000026.*(AlphaStall.^4));

    %Set values for Re = 3000
    Cl3000 =
0.0502+(0.521.*Alpha)+(-0.0277.*(Alpha.^2))+(0.00139.*(Alpha.^3))+(-0.0000242.*(A
lpha.^4));               %Sets up Cl Vector
    Cd3000 =
0.0668+(-0.00528*Alpha)+(0.00188*(Alpha.^2))+(-0.0000706.*(Alpha.^3))+(0.00000084
.*(Alpha.^4));               %Sets up Cd Vector
    Cdp3000 =
0.04+(-0.0157.*Alpha)+(0.00385.*(Alpha.^2))+(-0.00019.*(Alpha.^3))+(0.00000317.*(
Alpha.^4));               %Sets up Cdp Vector
    Cls3000 =
0.0502+(0.521.*AlphaStall)+(-0.0277.*(AlphaStall.^2))+(0.00139.*(AlphaStall.^3))+
(-0.0000242.*(AlphaStall.^4));
```

```
    Cds3000 =
0.0668+(-0.00528*AlphaStall)+(0.00188*(AlphaStall.^2))+(-0.0000706.*(AlphaStall.^
3))+(0.00000084.*(AlphaStall.^4));

    %Set values for Re = 5000
    Cl5000 =
-0.0264+(0.375.*Alpha)+(-0.0674.*(Alpha.^2))+(0.00635.*(Alpha.^3))+(-0.000312.*(A
lpha.^4))+(0.00000774.*(Alpha.^5))+(-0.0000000764.*(Alpha.^6));
%Sets up Cl Vector
    Cd5000 =
0.0553+(-0.00684*Alpha)+(0.00224*(Alpha.^2))+(-0.0000895.*(Alpha.^3))+(0.00000116
.*(Alpha.^4));
    Cdp5000 =
0.034+(-0.0131.*Alpha)+(0.00336.*(Alpha.^2))+(-0.000153.*(Alpha.^3))+(0.00000231.
*(Alpha.^4));
    Cls5000 =
-0.0264+(0.375.*AlphaStall)+(-0.0674.*(AlphaStall.^2))+(0.00635.*(AlphaStall.^3))
+(-0.000312.*(AlphaStall.^4))+(0.00000774.*(AlphaStall.^5))+(-0.0000000764.*(Alph
aStall.^6));
    Cds5000 =
0.0553+(-0.00684*AlphaStall)+(0.00224*(AlphaStall.^2))+(-0.0000895.*(AlphaStall.^
3))+(0.00000116.*(AlphaStall.^4));

    %Loop for interpolating the lift and drag values
    %Use Cl(n,1) here so on
    %May have to change the (n,1) for the vectors in the function
    n = 1;                              %Sets first variable for the loop
    while n <= Ni
        if Re(n) <= 1000
            Cl(n,1) = Interpolate(0,Re(n),1000,0,Cl1000(n));
            Cd(n,1) = Interpolate(0,Re(n),1000,0,Cd1000(n));
            Cdp(n,1) = Interpolate(0,Re(n),1000,0,Cdp1000(n));
            Cls(n,1) = Interpolate(0,Re(n),1000,0,Cls1000(n));
            Cds(n,1) = Interpolate(0,Re(n),1000,0,Cds1000(n));
```

```
    elseif Re(n) > 1000 && Re(n) <= 3000
        Cl(n,1) = Interpolate(1000,Re(n),3000,Cl1000(n),Cl3000(n));
        Cd(n,1) = Interpolate(1000,Re(n),3000,Cd1000(n),Cd3000(n));
        Cdp(n,1) = Interpolate(1000,Re(n),3000,Cdp1000(n),Cdp3000(n));
        Cls(n,1) = Interpolate(1000,Re(n),3000,Cls1000(n),Cls3000(n));
        Cds(n,1) = Interpolate(1000,Re(n),3000,Cds1000(n),Cds3000(n));
    elseif Re(n) > 3000
        Cl(n,1) = Interpolate(3000,Re(n),5000,Cl3000(n),Cl5000(n));
        Cd(n,1) = Interpolate(3000,Re(n),5000,Cd3000(n),Cd5000(n));
        Cdp(n,1) = Interpolate(3000,Re(n),5000,Cdp3000(n),Cdp5000(n));
        Cls(n,1) = Interpolate(3000,Re(n),5000,Cls3000(n),Cls5000(n));
        Cds(n,1) = Interpolate(3000,Re(n),5000,Cds3000(n),Cds5000(n));
    end
    n = n+1;
end

%First calculate the your custom correction, then the Corrigan
%Schilling Correction
b1 = 1.43656968592719;
b2 = 1.75545378442278;
b3 = 0.518367841387982;
b4 = 2.14917276732139;
b5 = -2.41326118040382;
b6 = 1.85996056350822;
b7 = 3.26463808395347;
b8 = -2.96910717977840;
b9 = 2.28183150338262;
b10 = -1.10158216973537;
b11 =  0.06048150937307982;
b12 = 0.02;
b13 =  0.07;

A = (Alpha - Alphazi)./(AlphaStall-Alphazi);
```

```
    fAlpha = (sin((b1.*A) + b2).*sin((b3.*A) + b4).*sin((b5.*A) +
b6).*sin((b7.*A) + b8).*sin((b9.*A) + b10))+1;
    Cl = b11.*(Re.^b12).*(0.01.^b13).*fAlpha.*Cl;
    Cl = Cl + (DeltaAlpha.*AlphaSlope);
    fAlpha = (sin(b1+b2).*sin(b3+b4).*sin(b5+b6).*sin(b7+b8).*sin(b9+b10))+1;
    Cls = b11.*(Re.^b12).*(0.01.^b13).*fAlpha.*Cls;      %.*(0.01.^b13)
    Cls = Cls + (DeltaAlpha.*AlphaSlope);


    %Here do the post-stall Viterna-Corrigan correction
    %First calculate A1, B1, etc. then loop for rest

    B1 =  (1+(0.065*AR))/0.91; %1.11+0.018*AR; %(1+(0.065*AR))/0.91;
    A1 = B1/2;
    A2 = Cls-(B1.*sin(AlphaStallRad).*cos(AlphaStallRad));
    B2 = Cds-((B1.*(sin(AlphaStallRad).^2))./cos(AlphaStallRad));

    n = 1;
    while n <= Ni
      if Alpha(n)> AlphaStall(n)
          Cl(n,1) =
(A1.*sin(2.*AlphaRad(n)))+(A2(n,1).*((cos(AlphaRad(n)).^2)./sin(AlphaRad(n))));
          Cd(n,1) = (B1.* (sin(AlphaRad(n)).^2)) + (B2(n,1).*cos(AlphaRad(n)));
      end
      n = n + 1;
    end
end

%This is the function you'll use to make your interpolater function
function y_med = Interpolate (x_low,x_med,x_high,y_low,y_high)
   y_med = (((x_med-x_low)/(x_high-x_low))*(y_high-y_low))+y_low;
end

%This code was used to create the coefficient correction using imported
```

```
%data. Cd corrections failed and were commented out

ClCorrect = ClCorrectionData(:,5);
CdCorrect = CdCorrectionData(:,5);
beta0 = [1,1,0.1,-1.501,1,-1.501,1, 0.0001,1,1,1,-0.15,-1];
ClCorrectionMod = fitnlm(ClCorrectionData,ClCorrect,@CoeffCorrection,beta0);   %Creates
our Thrust Model
%CdCorrectionMod = fitnlm(CdCorrectionData,CdCorrect,@CoeffCorrection,beta0);   %Creates
our Thrust Model

bvalueCl = table2array(ClCorrectionMod.Coefficients);                           %Gets our
coefficients out of our Model
yhatCl = CoeffCorrection(bvalueCl(:,1),ClCorrectionData(:,(1:8)));             %Finds the
model values from our Master Matrix
%bvalueCd = table2array(CdCorrectionMod.Coefficients);                           %Gets
our coefficients out of our Model
%yhatCd = CoeffCorrection(bvalueCd(:,1),CdCorrectionData(:,(1:6)));             %Finds
the model values from our Master Matrix

RMSECl = sqrt(mean((ClCorrectionData(:,6)-ClCorrect).^2));
RMSEClnew = sqrt(mean((yhatCl-ClCorrect).^2));
%RMSECd = sqrt(mean((CdCorrectionData(:,6)-CdCorrect).^2));
%RMSECdnew = sqrt(mean((yhatCd-CdCorrect).^2));

figure
plot(ClCorrectionData(:,1),ClCorrect,'o');
pause(1);
hold on
plot(ClCorrectionData(:,1),yhatCl,'o');
hold on
pause(1);
plot(ClCorrectionData(:,1),ClCorrectionData(:,6),'o');
title('Model vs. Actual for Cd Coefficients ')
xlabel('Reynolds Number')
ylabel('Cd Coeff')
legend('Experimental Data','XFLR5 + Correction Model', 'XFLR5 Data')
hold off
```

```
figure
plot(ClCorrectionData(1:18,4),ClCorrect(1:18,1),'-o');
pause(1);
hold on
plot(ClCorrectionData(1:18,4),yhatCl(1:18,1),'-d');
hold on
pause(1);
plot(ClCorrectionData(1:18,4),ClCorrectionData(1:18,6),'-*');
%title('E61 @ Re = 17,000')
xlabel('Angle of Attack')
ylabel('Cl Coeff')
legend('Experimental Data','XFLR5 + Correction Model', 'XFLR5 Data')
hold off

figure
plot(ClCorrectionData(38:55,4),ClCorrect(38:55,1),'-o');
pause(1);
hold on
plot(ClCorrectionData(38:55,4),yhatCl(38:55,1),'-d');
hold on
pause(1);
plot(ClCorrectionData(38:55,4),ClCorrectionData(38:55,6),'-*');
%title('GOE 795 @ Re = 17,000')
xlabel('Angle of Attack')
ylabel('Cl Coeff')
legend('Experimental Data','XFLR5 + Correction Model', 'XFLR5 Data')
hold off

figure
plot(ClCorrectionData(174:194,4),ClCorrect(174:194,1),'-o');
pause(1);
hold on
plot(ClCorrectionData(174:194,4),yhatCl(174:194,1),'-d');
hold on
pause(1);
plot(ClCorrectionData(174:194,4),ClCorrectionData(174:194,6),'-*');
```

```
%title('GOE 801 @ Re = 21,000')
xlabel('Angle of Attack')
ylabel('Cl Coeff')
legend('Experimental Data','XFLR5 + Correction Model', 'XFLR5 Data')
hold off


function yhat = CoeffCorrection(beta,x)
    b1 = beta(1);
    b2 = beta(2);
    b3 = beta(3);
    b4 = beta(4);
    b5 = beta(5);
    b6 = beta(6);
    b7 = beta(7);
    b8 = beta(8);
    b9 = beta(9);
    b10 = beta(10);
    b11 = beta(11);
    b12 = beta(12);
    b13 = beta(13);
    Re = x(:,1);
    T = x(:,2);
    Alpha = x(:,4);
    Alphaz = x(:,7);
    Alphas = x(:,8);
    Alpha = (Alpha - Alphaz)./(Alphas-Alphaz);
    Cl = x(:,6);

    fAlpha = (sin((b1.*Alpha) + b2).*sin((b3.*Alpha) + b4).*sin((b5.*Alpha) +
b6).*sin((b7.*Alpha) + b8).*sin((b9.*Alpha) + b10))+1;
    yhat = b11.*(Re.^b12).*(T.^b13).*fAlpha.*Cl;
end

%This code recreates the previously done experiment and compares the
%corrected and uncorrected results
```

```
% initialization
%clear all;
%clc;
%close all;

%Input Variables

Ni = 500;                          %Number of blade sections
RPM = 3000:200:4000;               %Blade Rotation
Nb = 2;                            %Number of Blades
R = 0.75;                          %Total Radius (ft)
Rhub = 0.0625;                     %Hub Radius (ft)
h = 0;                             %Assumed Height (ft)
V = 0;                             %Forward Speed (ft/s)
E = 9e9;
AlphaIdeal = [18,28,38];                  %Angle at Hub (deg)

%Creating blank matrices for our output values

T = zeros(length(AlphaIdeal)*length(RPM),1);      %Thurst Sim Data
P = zeros(length(AlphaIdeal)*length(RPM),1);      %Power Sim Data
Alpha = zeros(length(AlphaIdeal)*length(RPM),1);  %Alpha Values
Omega = zeros(length(AlphaIdeal)*length(RPM),1);  %RPM Values
l = 1;  %Total pass through counter
m = 1;  %RPM pass through counter
while m <= length(AlphaIdeal)
    n = 1;                  %Radial Pass throughs, reset each RPM
    while n <= length(RPM)
        Alpha(l) = Alpha(m);
        Omega(l) = Omega(n);
        [T(l), P(l), Q, Eff, MachMax, Comp, ReMax] =
Prop(Ni,RPM(n),Nb,R,Rhub,h,V,AlphaIdeal(m),E);   %Calls the BEMT Function
        l = l + 1;
        n = n + 1;
```

```
    end
    m = m + 1;
end

%Conversions
T = 4.44822162.*T;   %Converts Thrust to N
P = 745.699872.*P;   %Converts Power to Watts

%Experimental and uncorrected data
Texp =
[0.1,0.12,0.165,0.19,0.2,0.21;0.19,0.225,0.27,0.305,0.33,0.365;0.33,0.37,0.42,0.4
75,0.52,0.57];
Pexp =
[2.4,2.5,3,3.5,4.1,4.8;4.2,4.9,5.8,6.8,7.5,8.2;8.5,10.2,12.1,14,15.8,17.9];
Tun =
[1.40,1.63,1.85,2.04,2.20,2.33;2.11,2.45,2.75,3.02,2.87,3.03;0.58,0.57,0.48,0.31,
0.053,-0.30];
Pun =
[16.55,19.72,22.85,25.91,28.87,31.70;25.42,30.04,34.54,38.90,58.27,63.39;7.66,9.8
5,12.32,15.11,18.24,21.76];

%Graphing Results at 18 deg
figure
subplot(1,2,1);
plot(RPM,Texp(1,:),'-o',RPM,Tun(1,:),'-*',RPM,T(1:6),'-d');
%title('Thrust Comparison @ Beta = 18 deg')
xlabel('RPM')
ylabel('Thrust (N)')
legend('Data from experiment','Uncorrected Simulation','Corrected Simulation')

subplot(1,2,2);
plot(RPM,Pexp(1,:),'-o',RPM,Pun(1,:),'-*',RPM,P(1:6),'-d');
%title('Power Comparison @ Beta = 18 deg')
xlabel('RPM')
```

```
ylabel('Power (W)')
%legend('Data from experiment','Uncorrected Simulation','Corrected Simulation')

%Graphing Results at 28 deg
figure
subplot(1,2,1);
plot(RPM,Texp(2,:),'-o',RPM,Tun(2,:),'-*',RPM,T(7:12),'-d');
%title('Thrust Comparison @ Beta = 28 deg')
xlabel('RPM')
ylabel('Thrust (N)')
legend('Data from experiment','Uncorrected Simulation','Corrected Simulation')

subplot(1,2,2);
plot(RPM,Pexp(2,:),'-o',RPM,Pun(2,:),'-*',RPM,P(7:12),'-d');
%title('Power Comparison @ Beta = 28 deg')
xlabel('RPM')
ylabel('Power (W)')
%legend('Data from experiment','Uncorrected Simulation','Corrected Simulation')

%Graphing Results at 38 deg
figure
subplot(1,2,1);
plot(RPM,Texp(3,:),'-o',RPM,Tun(3,:),'-*',RPM,T(13:18),'-d');
%title('Thrust Comparison @ Beta = 38 deg')
xlabel('RPM')
ylabel('Thrust (N)')
legend('Data from experiment','Uncorrected Simulation','Corrected Simulation')

subplot(1,2,2);
plot(RPM,Pexp(3,:),'-o',RPM,Pun(3,:),'-*',RPM,P(13:18),'-d');
%title('Power Comparison @ Beta = 38 deg')
xlabel('RPM')
ylabel('Power (W)')
%legend('Data from experiment','Uncorrected Simulation','Corrected Simulation')
```

```
%This code creates a large amount of BEMT Data and fits it to a simpler
%function

% initialization
clc;
close all;

%Input Variables

Ni = 100;                           %Number of blade sections
RPM = 3000;                         %Blade Rotation
Nb = 2;                             %Number of Blades
R = 0.75:0.1:1.75;                      %Total Radius (ft)
Rhub = 0.1*R;                       %Hub Radius (ft)
h = 0;                              %Assumed Height (ft)
V = 0;                              %Forward Speed (ft/s)
AlphaIdeal = 5:1:25;               %Angle at Hub (deg)
E = 9e9;                            %Youngs Stuff
CF = 2.4:0.05:2.8;                     %Chord Factor
TMin = 0.8;                         %Thrust minimum (lbf)
MMax = 0.7;                         %Mach number maximum

%Creating blank matrices for our output values
MastMat = zeros(length(R)*length(AlphaIdeal),7);  %Master Matrix
l = 1;  %Total pass through counter
m = 1;  %RPM pass through counter
while m <= length(R)
    n = 1;
    while n <= length(AlphaIdeal)
        o = 1;
        while o <= length(CF)
            MastMat(l,(1:3))=[R(m),AlphaIdeal(n),CF(o)];
```

```
            [MastMat(l,4), P, Q, MastMat(l,5), MachMax, Comp, ReMax] =
Prop(Ni,RPM,Nb,R(m),Rhub(m),h,V,AlphaIdeal(n),E,CF(o));
            l = l+1;
            o = o+1;
        end
        n = n+1;
    end
    m = m+1;
end

%Outlier Identification
l = length(AlphaIdeal)*length(CF);                          %Length of sections
we're checking for outliers
a = 1;                                                      %Starting Index
%Thrust Outliers
while a <= length(R)
    m = (l*(a-1))+1;                                        %Front Index
    n = a*l;                                                %Back Index
    MastMat((m:n),6) = isoutlier(MastMat((m:n),4));         %Identifies Thrust
Outliers
    a = a+1;
end
a = 1;                                                      %Resets Index
%Efficiency Outliers
while a <= length(R)
    m = (l*(a-1))+1;                                        %Front Index
    n = a*l;                                                %Back Index
    MastMat((m:n),7) = isoutlier(MastMat((m:n),5));         %Identifies Efficiency
Outliers
    a = a+1;
end

%Outlier Elimination
m = 1;                                                  %Index
```

```
while m <= length(MastMat)
    if MastMat(m,6) == 1 || MastMat(m,7) == 1   %Identifies Thrust/Efficiency
Outliers
        MastMat(m,:) = [];                      %Removes outliers row
    else
    m = m+1;
    end
end

%Model Generation
beta0 = [-1, 1, 1, 1, 1, 1, 1, 1, 1];                           %Coefficient
Guess Vector
ThrustMod = fitnlm(MastMat(:,(1:3)),MastMat(:,4),@ThrustModel,beta0);   %Creates
our Thrust Model
bvalue = table2array(ThrustMod.Coefficients);                           %Gets our
coefficients out of our Model
yhat1 = ThrustModel(bvalue(:,1),MastMat(:,(1:3)));                      %Finds
the model values from our Master Matrix



beta0 = [-1, 1, 1, 1, 1, 1, 1, 1, 1];                           %Coefficient
Guess Vector
EffMod = fitnlm(MastMat(:,(1:3)),MastMat(:,5),@ThrustModel,beta0);      %Creates
our Efficiency Model
bvalue = table2array(EffMod.Coefficients);                              %Gets our
coefficients out of our Model
yhat2 = ThrustModel(bvalue(:,1),MastMat(:,(1:3)));                      %Finds
the model values from our Master Matrix
```

```
figure
subplot(1,2,1)
plot(MastMat(:,1),MastMat(:,4),'o')
title('BEMT Thrust Values')
xlabel('Radius (ft)','fontsize',14)
ylabel('Thrust (lb)','fontsize',14)
axis ([0.6 1.8 0 0.5]);
subplot(1,2,2)
plot(MastMat(:,1),yhat1,'o')
title('Modeled Thrust Values')
xlabel('Radius (ft)','fontsize',14)
ylabel('Thrust (lb)','fontsize',14)
axis ([0.6 1.8 0 0.5]);
figure
subplot(1,2,1)
plot(MastMat(:,1),MastMat(:,5),'o')
title('BEMT Efficiency Values')
xlabel('Radius (ft)','fontsize',14)
ylabel('Efficiency (lb/BHP)','fontsize',14)
axis ([0.6 1.8 0 8]);

subplot(1,2,2)
plot(MastMat(:,1),yhat2,'o')
title('Modeled Efficiency Values')
xlabel('Radius (ft)','fontsize',14)
ylabel('Efficiency (lb/BHP)','fontsize',14)
axis ([0.6 1.8 0 8]);

%Model used for Thrust and Efficiency (Outdated Name)
function yhat = ThrustModel(beta,x)
    b1 = beta(1);
    b2 = beta(2);
    b3 = beta(3);
    b4 = beta(4);
```

```
    b5 = beta(5);
    b6 = beta(6);
    b7 = beta(7);
    b8 = beta(8);
    b9 = beta(9);
    x1 = x(:,1);
    x2 = x(:,2);
    x3 = x(:,3);
    yhat =
(b1+(b2.*x2)+(b3.*(x2.^2))+(b4.*(x2.^3))+(b5.*(x2.^4))+(b6.*(x2.^5))+(b7.*(x2.^6)
)).*(x3.^b8).*(x1.^b9);
end

%NSGA used to optimize chord, pitch, and radius

%Initial settings
NPop = 200;                          %Population Size
NVars = 3;                           %Number of variables
NObj = 2;                            %Number of Objective functions
Obj1Pos = NVars + 1;                 %Position of Objective Value 1
Obj2Pos = NVars + 2;                 %Position of Objective Value 2
RankPos = NVars + 3;                 %Rank Position
CDPos = NVars + 4;                   %Crowding Distance Position
DSPos = NVars + 5;                   %Domination Set Position
DCPos = NVars + 6;                   %Domination Count Position
Generations = 100;                   %Number of Generations
CrossRate = 0.5;                     %Cross over rate of parents
NPairs = round(CrossRate*NPop/2)*2;  %Number of Parents
MutRate = 0.3;                       %Mutation Rate
NMuts = round(MutRate*NPop);         %Number of Mutants
V1Min = 0.75;                         %Min value for variable 1
V1Max = 1.75;                          %Max value for variable 1
V2Min = 2.4;                         %Min value for variable 2
V2Max = 2.8;                          %Max value for variable 2
V3Min = 5;                          %Min value for variable 3
V3Max = 25;                          %Max value for variable 3
```

```
%V4Min = -20;                           %Min value for variable 4
%V4Max = 20;                            %Max value for variable 4

%Here we'll generate the necessary population
Pop = zeros(NPop,DCPos);

n = 1;
while n<=NPop
   Pop(n,1) = unifrnd(V1Min,V1Max);   %Fills our first variable
   Pop(n,2) = unifrnd(V2Min,V2Max);   %Fills the second variable
   Pop(n,3) = unifrnd(V3Min,V3Max);   %Fills our third variable
   %Pop(n,4) = unifrnd(V4Min,V4Max);   %Fills the fourth variable
   %if Constraint(Pop,n) == 1
       n=n+1;
   %else
   %    continue
   %end
end

[ObjArray1, ObjArray2] = ObjectiveFuncs(Pop);    %Generate objective values
Pop(:,Obj1Pos) = ObjArray1;                      %Generates values for objective 1
Pop(:,Obj2Pos) = ObjArray2;                      %Generates values for objective 2


[NRank1, Pop] = NonDomSort(Pop, NPop, Obj1Pos, Obj2Pos, RankPos, DSPos, DCPos);
%Sorts our current population
Pop = sortrows(Pop,RankPos);
%Organizes based on rank
Pop = CrowdDist(Pop,Obj1Pos,Obj2Pos,CDPos,NPop);
%Calculates Crowd Distancing
Pop = Pop(1:NPop,1:DCPos);

%Main Loop for generations
n=1;
while n<=Generations

%Plot generation
```

```
Rank1Obj1 = [];                                    %Blank Array for Obj Func 1
Rank1Obj2 = [];                                    %Blank Array for Obj Func 2
Rank1Thu1 = [];
Rank1Eff1 = [];

m = 1;
while m<=NPop
    if Pop(m,RankPos) == 1
        Rank1Obj1 = [Rank1Obj1 Pop(m,Obj1Pos)];   %Fills Obj Func 1 Array for Rank 1
        Rank1Obj2 = [Rank1Obj2 Pop(m,Obj2Pos)];   %Fills Obj Func 2 Array for Rank 1
        Rank1Thu1 = [Rank1Thu1 Pop(m,1)];
        Rank1Eff1 = [Rank1Eff1 Pop(m,2)];
    end
    m = m +1;
end

%subplot(2,1,1)
scatter(-1.*Rank1Obj1, -1.*Rank1Obj2, 'o');                    %Generates our plot based
on our generated arrays
%title('Criterion Space ')   %'Thrust and Efficiency Optimization Pareto Frontier'
xlabel('Thrust (lb)')                                          %'Thrust (lbf)'
ylabel('Efficiency (lb/BHP)')                                  %'Efficiency (lbf/BHP)'
%subplot(2,1,2)
%scatter(Rank1Thu1, Rank1Eff1);
pause(0.05);
n=n+1;

if n==Generations
     break;
end

%Next Gencreation
Pop2 = Mating(Pop, NPairs, NPop, RankPos, CDPos,DCPos);
%Creates Reproduced Population
[ObjArray1, ObjArray2] = ObjectiveFuncs(Pop2);
Pop2(:,Obj1Pos) = ObjArray1;                    %Generates values for objective 1
Pop2(:,Obj2Pos) = ObjArray2;                    %Generates values for objective 2
```

```
Pop3 = Mutate(Pop, NMuts, NPop, RankPos, CDPos, DCPos, V1Min, V2Min, V1Max, V2Max);
%Creates Mutated Population
[ObjArray1, ObjArray2] = ObjectiveFuncs(Pop3);
Pop3(:,Obj1Pos) = ObjArray1;                    %Generates values for objective 1
Pop3(:,Obj2Pos) = ObjArray2;                    %Generates values for objective 2

Pop=[Pop;
     Pop2;
     Pop3];
 CPop = NPop+NPairs+NMuts;
 i = 1;
 while i <=CPop
 Pop(i,RankPos) = 0;
 Pop(i,CDPos) = 0;
 Pop(i,DSPos) = 0;
 Pop(i,DCPos) = 0;
 i = i+1;
 end

[NRank1, Pop] = NonDomSort(Pop, CPop, Obj1Pos, Obj2Pos, RankPos, DSPos, DCPos);
%Sorts our current population
Pop = sortrows(Pop,RankPos);
%Organizes based on rank
Pop = CrowdDist(Pop,Obj1Pos,Obj2Pos,CDPos,NPop);
%Calculates Crowd Distancing
Pop = Pop(1:NPop,1:DCPos);

end



%Functions used in this program
function [ObjArray1, ObjArray2] = ObjectiveFuncs(Pop)

    ObjArray1 =
-1.*((Pop(:,1).^3.98342788584788)).*((Pop(:,2).^1.57451748766688)).*(-0.00264686292660979
+(0.00618332540848222.*Pop(:,3))+(-0.00121192584284126.*(Pop(:,3).^2))+(0.000111574693645
```

```
910.*(Pop(:,3).^3))+(-5.01689605443821e-06.*(Pop(:,3).^4))+(9.92805430873840e-08.*(Pop(:,
3).^5))+(-5.89884390432631e-10.*(Pop(:,3).^6)));
    ObjArray2 =
-1.*((Pop(:,1).^-1.14917251645953)).*((Pop(:,2).^0.235571247807191)).*(4.82635760560193+(
0.316809760738228.*Pop(:,3))+(-0.158409177057510.*(Pop(:,3).^2))+(0.0189128656993398.*(Po
p(:,3).^3))+(-0.00113856602869597.*(Pop(:,3).^4))+(3.48667688119601e-05.*(Pop(:,3).^5))+(
-4.25677537980454e-07.*(Pop(:,3).^6)));
end

function [Rank, x] = NonDomSort(Pop, NPop, Obj1Pos, Obj2Pos, RankPos, DSPos, DCPos)
    Rank = 0;
    n = 1;
    while n <=NPop - 1
       m = 1;
       while m <=NPop - 1
           if m == n
               m = m+1;
           end
           b = Dom(Pop, n, m, Obj1Pos, Obj2Pos);
           c = Dom(Pop, m, n, Obj1Pos, Obj2Pos);
           if b == 0
               Pop(n, DSPos) = m;
           elseif c == 0
               Pop(n, DCPos) = Pop(n, DCPos) + 1;
           end
           m = m+1;
       end
       if Pop(n, DCPos) == 0
           Pop(n, RankPos) = 1;
           Rank = Rank + 1;
       end
       n = n+1;
    end

    r = 2;
    while true
        Blank = [];
```

```
    i = 1;
    while i <= NPop
       Pop(i,DCPos) = Pop(i,DCPos)-1;
       if Pop(i,DCPos) == 0
           Pop(i,RankPos) = r;
           Blank = [Blank i];
       end
       i = i+1;
    end

    if isempty(Blank)
        break
    end

    F{r} = Blank;
    r = r+1;

end

n = 1;

while n<= NPop
    if Pop(n,RankPos) == 0
        Pop(n,RankPos) = 2000;
    end
    n = n+1;
end

x = Pop;

function a = Dom (Pop,n,m,Obj1Pos,Obj2Pos)
    d = Pop(n,Obj1Pos)<=Pop(m, Obj1Pos) && Pop(n,Obj2Pos) <= Pop(m, Obj2Pos);
    f = Pop(n,Obj1Pos) < Pop(m, Obj1Pos) || Pop(n,Obj2Pos) < Pop(m, Obj2Pos);
    if d
        if f
           a = 0;
```

```
            else
                a = 1;
            end
        else
            a = 1;
        end
    end
end

function Muts = Mutate(Pop, NMuts, NPop, RankPos, CDPos, DCPos, V1Min, V2Min, V1Max,
V2Max)

         Muts = zeros(NMuts,DCPos);
         n=1;
         while n <=NMuts
             P1 = select(Pop, NPop, RankPos, CDPos);
             f1 = 0.01*randi([75 175], 1);
             f2 = 0.01*randi([240 280], 1);
             f3 = 0.1*randi([50 250], 1);
             %f4 = randi([V4Min V4Max], 1);
             Muts(n,1) = (0.001*f1)+P1(1);
             Muts(n,2) = (0.001*f2)+P1(2);
             Muts(n,3) = (0.001*f3)+P1(3);
             %Muts(n,4) = (0.001*f4)+P1(4);
             %if Constraint(Muts,n) == 1
                n=n+1;
             %else
             %   continue
             %end
         end
end

function x = CrowdDist(Pop,Obj1Pos,Obj2Pos,CDPos, NPop)
   n=1;
   while n <= NPop
       CrowdDist1 = 0;
       m=1;
```

```
        while m <= NPop
            CrowdDist1 = CrowdDist1 +
sqrt(((Pop(n,Obj1Pos)-Pop(m,Obj1Pos))^2)+((Pop(n,Obj2Pos)-Pop(m,Obj2Pos))^2));
            m = m+1;
        end
        Pop(n,CDPos) = CrowdDist1;
        n = n+1;
    end
 x = Pop;
end

function NewGen = Mating(Pop, NPairs, NPop, RankPos, CDPos,DCPos)

     NewGen = zeros(NPairs,DCPos);
     n=1;
     while n <=NPairs
        p1 = select(Pop, NPop, RankPos, CDPos);
        p2 = select(Pop, NPop, RankPos, CDPos);
        New1 = (p1+p2)/2;
        NewGen(n,1) = New1(1);
        NewGen(n,2) = New1(2);
        NewGen(n,3) = New1(3);
        %NewGen(n,4) = New1(4);
        %if Constraint(NewGen,n) == 1
           n=n+1;
        %else
        %   continue
        %end
     end
end

function XY = select(Pop, NPop, RankPos, CDPos)
     i=randi(NPop,1,2);
     if Pop(i(1),RankPos)<Pop(i(2),RankPos)
          XY = Pop(i(1),1:3);
     elseif Pop(i(1),RankPos)>Pop(i(2),RankPos)
         XY = Pop(i(2),1:3);
```

```
     else
        if Pop(i(1),CDPos)>Pop(i(2),CDPos)
             XY = Pop(i(1),1:3);
        else
             XY = Pop(i(2),1:3);
        end
     end
end

function a = Constraint(Pop,n)
if (-5*Pop(n,1))+Pop(n,2)-(Pop(n,3)^2) <= 0
    a = 1;
else
    a = 0;
end

end

%Preforms transformations on Airfoil Coord, exports results to .txt
    %Set up calculations
    Ni = 21;                            %Number of cross sections
    ID = 1:1:Ni;                        %Blade Section Vector
    ID = ID';                           %Transposes it for easier use
    R = 16;                              %Radius we're giving the object
    Rhub = 0.1*R;
    V = 0;                              %Assumes hover conditions
    RPM = 3000;
    AlphaIdeal = 6;
    w = zeros(Ni,1);                    %Initial Induced Veclocity (ft/s)
    w = w+1;                            %Gives w a value of 1
    DeltaR = (R-Rhub)/Ni;               %Hub Widths (ft)
    n = RPM/60;                         %Revolutions in (rev/s)
    Omega = RPM*pi/30;                  %Radial Speed (rad/s)
    r = (0:0.05:1).*R;                       %Sets our initial radial position
vectors
    r = r';
```

```
    CNum = length(S1223Mod3Coord);

    %Calculations of Key Values

    x = r./R;                                           %Radial Station as a
Percentage
    c = 2.8*R*(-0.101+(2.27.*x)+(-7.14.*(x.^2))+(8.38.*(x.^3))+(-3.37.*(x.^4)));
    %c = R*(-0.289+(4.91.*x)+(-13.6.*(x.^2))+(14.6.*(x.^3))+(-5.45.*(x.^4)));
%Chord Length by Radial Station (ft)
    ctr = c./r;                                         %Chord to radial station
ratio
    Omegar = Omega*r;                                   %Gives Radial Station
Linear Velocities (ft/s)
    Vr = sqrt((Omegar.^2)+(V^2));                       %Air Relative Speed (ft/s)
    PhiRad = atan(V./Omegar);                           %Phi (rad)
    Phi = PhiRad*180/pi;                                %Phi (deg)
    Beta = AlphaIdeal + Phi;                            %Beta (deg)
    BetaRad = -Beta*pi/180;                             %Beta (rad)
    Shift = c(3)*cos(BetaRad)*0.88069*0.5-(0.33*c(3));

    n = 1;

    while n <= Ni
       FN = ["HighThrustRadStat",num2str(100*x(n)),".txt"];
       FileName = join(FN,"");
       XSectCoordScaled = zeros(CNum,3);
       XSectCoordFinal = zeros(CNum,3);
       XSectCoordScaled(:,(2:3)) = S1223Mod3Coord(:,(2:3))*c(n);
       XSectCoordFinal(:,2) =
(XSectCoordScaled(:,2).*cos(BetaRad(n)))-(XSectCoordScaled(:,3).*sin(BetaRad(n)))
-(0.33*c(n)) - Shift(n);
       XSectCoordFinal(:,3) =
(XSectCoordScaled(:,2).*sin(BetaRad(n)))+(XSectCoordScaled(:,3).*cos(BetaRad(n)))
;
```

```
    XSectCoordFinal(:,1) = S1223Mod3Coord(:,1)*r(n);
    XSectCoord = array2table(XSectCoordFinal);
    writetable(XSectCoord,FileName,'WriteVariableNames',false);

    n = n + 1;
end
```

BIOGRAPHY OF THE AUTHOR

Ben Hebert was born in Palm Beach Gardens, FL on August 19th, 1995. He was raised in South Berwick, ME and graduated from Marshwood High School in 2013. He attended the University of Maine and graduated with a B.S. in Engineering Physics and a minor in Mathematics in 2017. He remained at the University of Maine and entered the Engineering Physics graduate program in the fall of that same year. After receiving his degree, Ben has plans for the future. Ben is a candidate for the Master of Engineering degree in Engineering Physics from the University of Maine in December 2020.